adafruit learning system

Adafruit PiTFT - 2.8" Touchscreen Display for Raspberry Pi

Created by lady ada

Last updated on 2015-05-18 01:00:16 PM EDT

Guide Contents

Guide Contents 2
Overview 4
Assembly 8
Easy Install 15
Ready to go image 15
DIY Installer script 15
Detailed Installation 20
Before you start 21
Download & Install Kernel 21
Touchscreen Install & Calibrate 31
Setting up the Touchscreen 31
Running evtest 33
AutoMagic Calibration Script 34
Manual Calibration 36
X Calibration 38
Using the Console 42
Turn off Console Blanking 45
Playing Videos 47
How To Play Videos 47
Converting/Resizing Videos 49
Displaying Images 53
Backlight Control 55
Controlling the backlight 55
Extras! 57
Tactile switch as power button 57
Making it easier to click icons in X 58
Boot to X Windows on PiTFT 59
Gesture Input 61
- Installation 61
- Usage 61

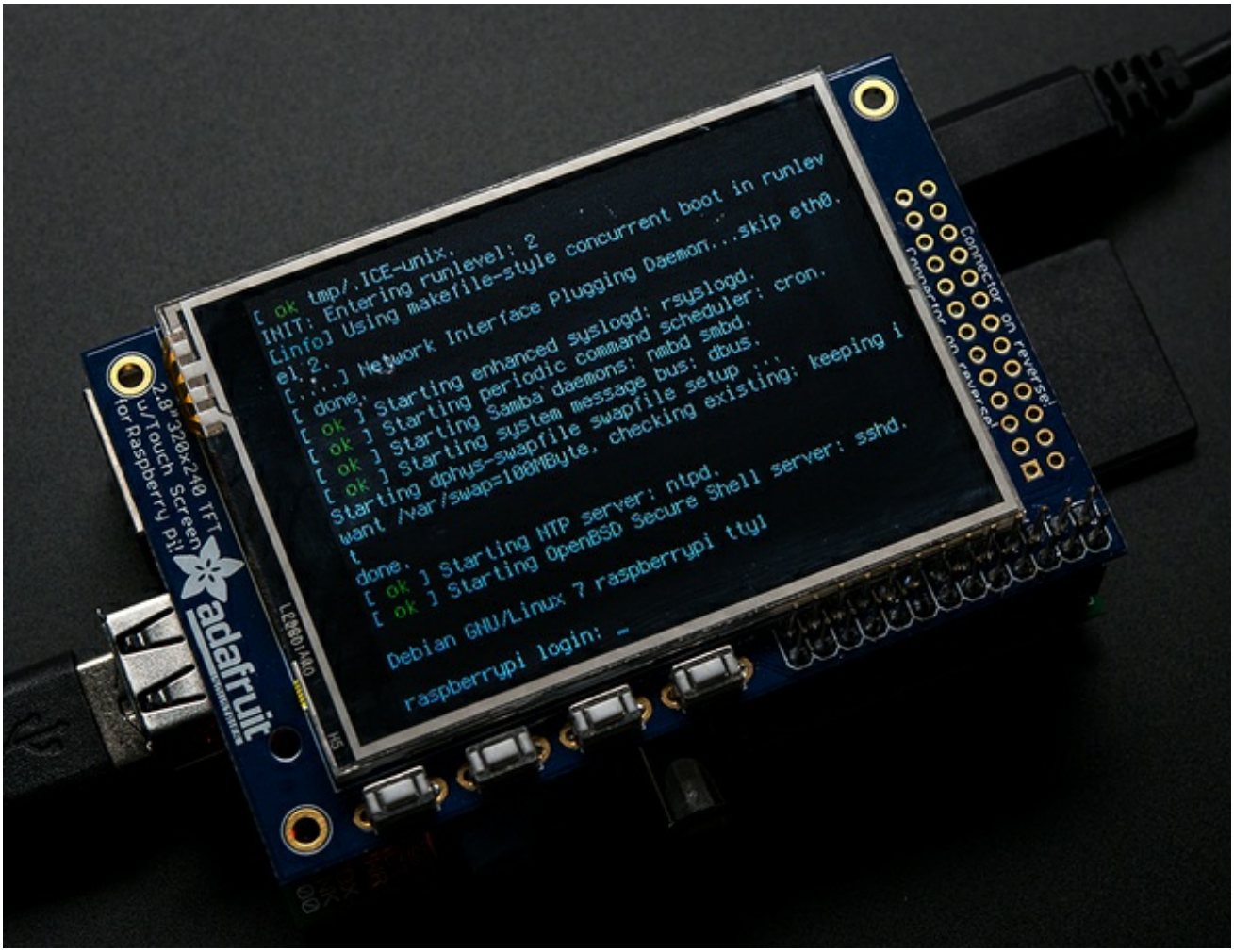
FAQ 65

Downloads 69
Schematic & Layout 69
Schematic & Layout 71

Overview

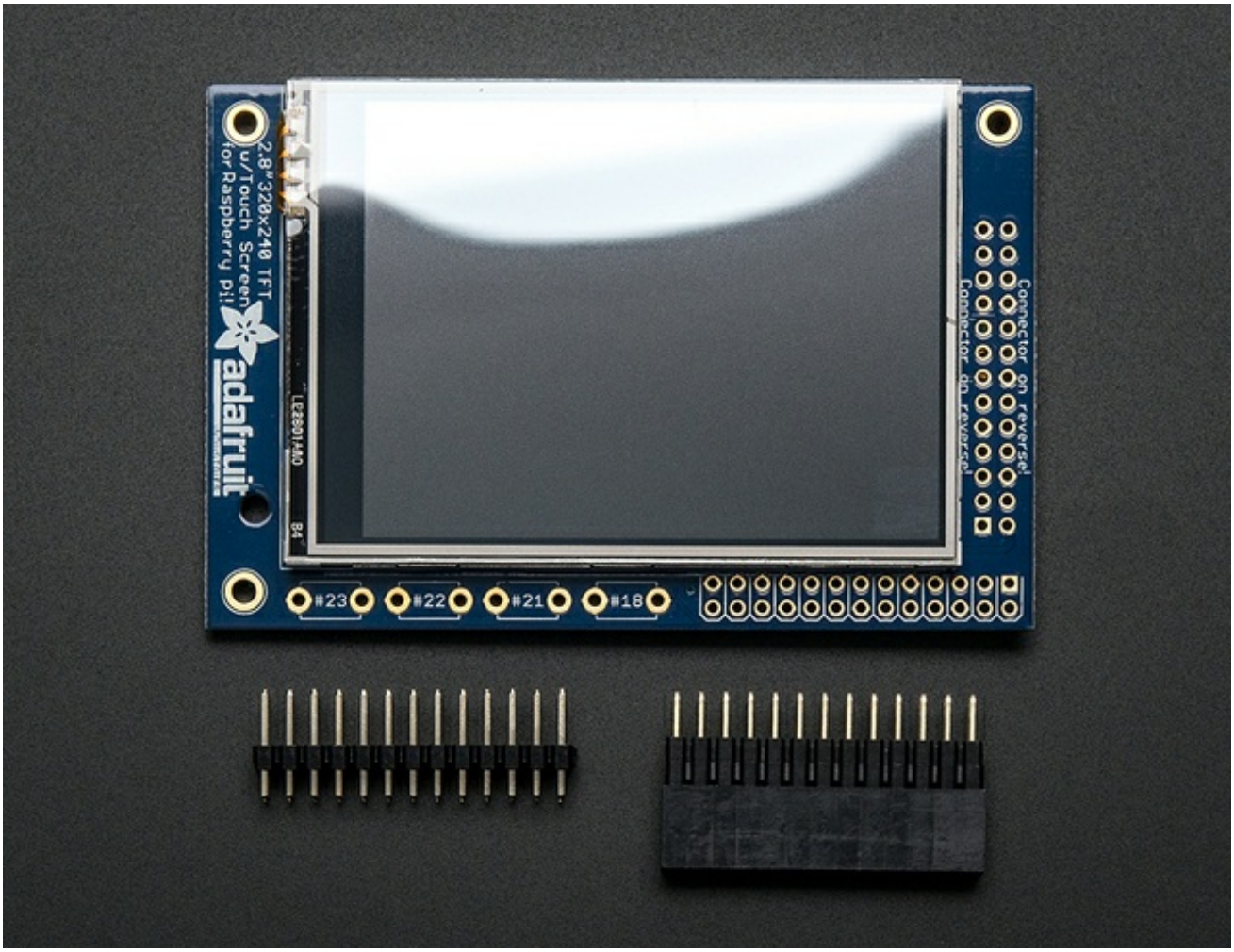
Is this not the cutest little display for the Raspberry Pi? It features a 2.8" display with 320x240 16-bit color pixels and a resistive touch overlay. The plate uses the high speed SPI interface on the Pi and can use the mini display as a console, X window port, displaying images or video etc. Best of all it plugs right in on top!

It's designed to fit nicely onto the Pi Model A or B but also works perfectly fine with the Pi 2 or Model B+ as long as you don't mind the PCB overhangs the USB ports by 5mm

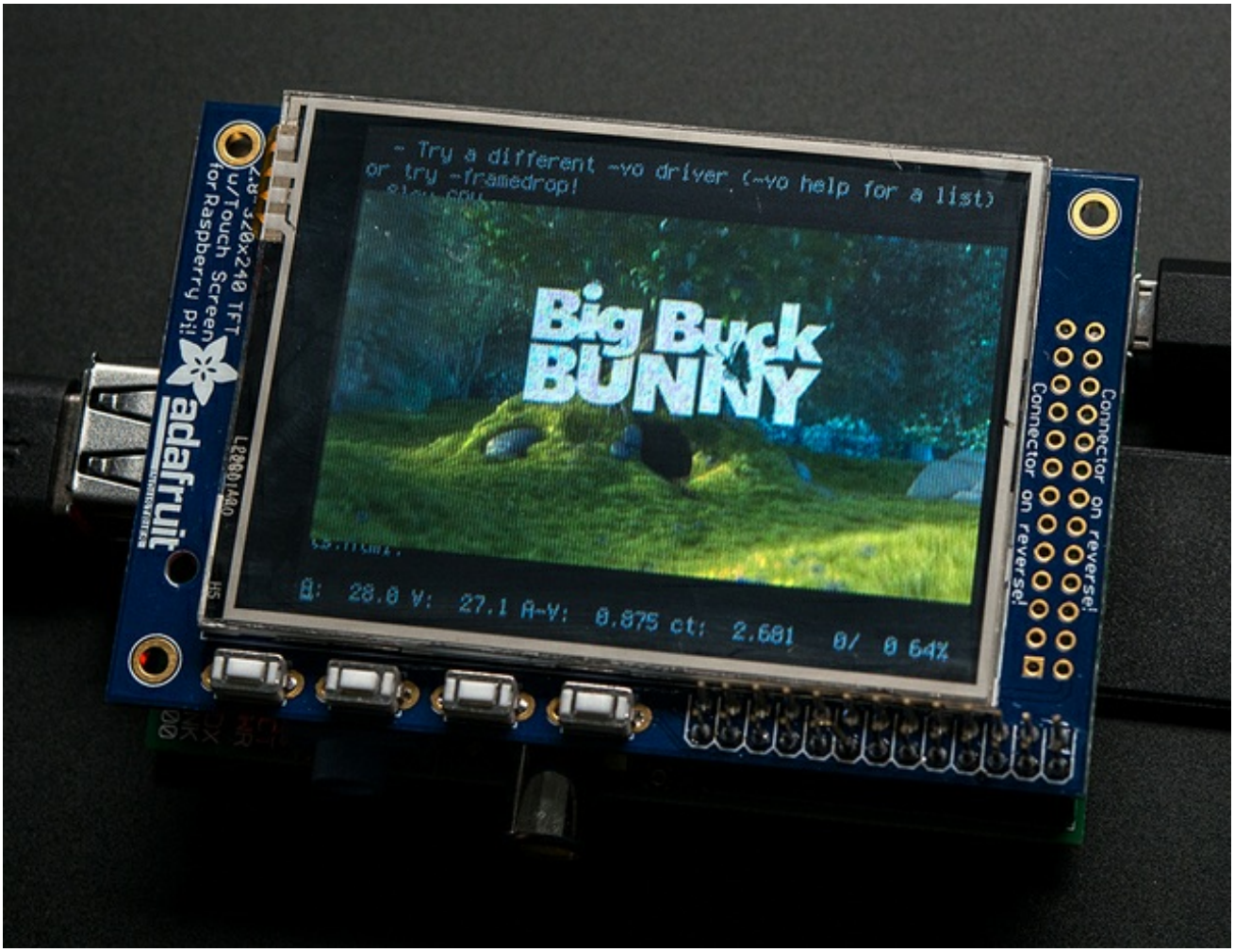

[ok] tmp/.ICE-unix.
INIT: Entering runlevel: 2
[info] Using makefile-style concurrent boot in runlev
el 2.
[....] Network Interface Plugging Daemon...skip eth0.
[done.
[ok] Starting enhanced syslogd: rsyslogd.
[ok] Starting periodic command scheduler: cron.
[ok] Starting Samba daemons: nmbd smbd.
[ok] Starting system message bus: dbus.
Starting dphys-swapfile swapfile setup ...
want /var/swap=100MByte, checking existing: keeping i
t
done.
[ok] Starting NTP server: ntpd.
[ok] Starting OpenBSD Secure Shell server: sshd.
Debian GNU/Linux 7 raspberrypi tty1
raspberrypi login: _
2.8" 320x240 TFT w/Touch Screen for Raspberry Pi!
adafruit
Connector on reverse!

This design uses the hardware SPI pins (SCK, MOSI, MISO, CE0, CE1) as well as GPIO #25 and #24. All other GPIO are unused. Since we had a tiny bit of space, there's 4 spots for optional slim tactile switches wired to four GPIOs, that you can use if you want to make a basic user interface. For example, you can use one as a power on/off button.

We bring out GPIO #23, #22, #21, and #18 to the four switch locations!



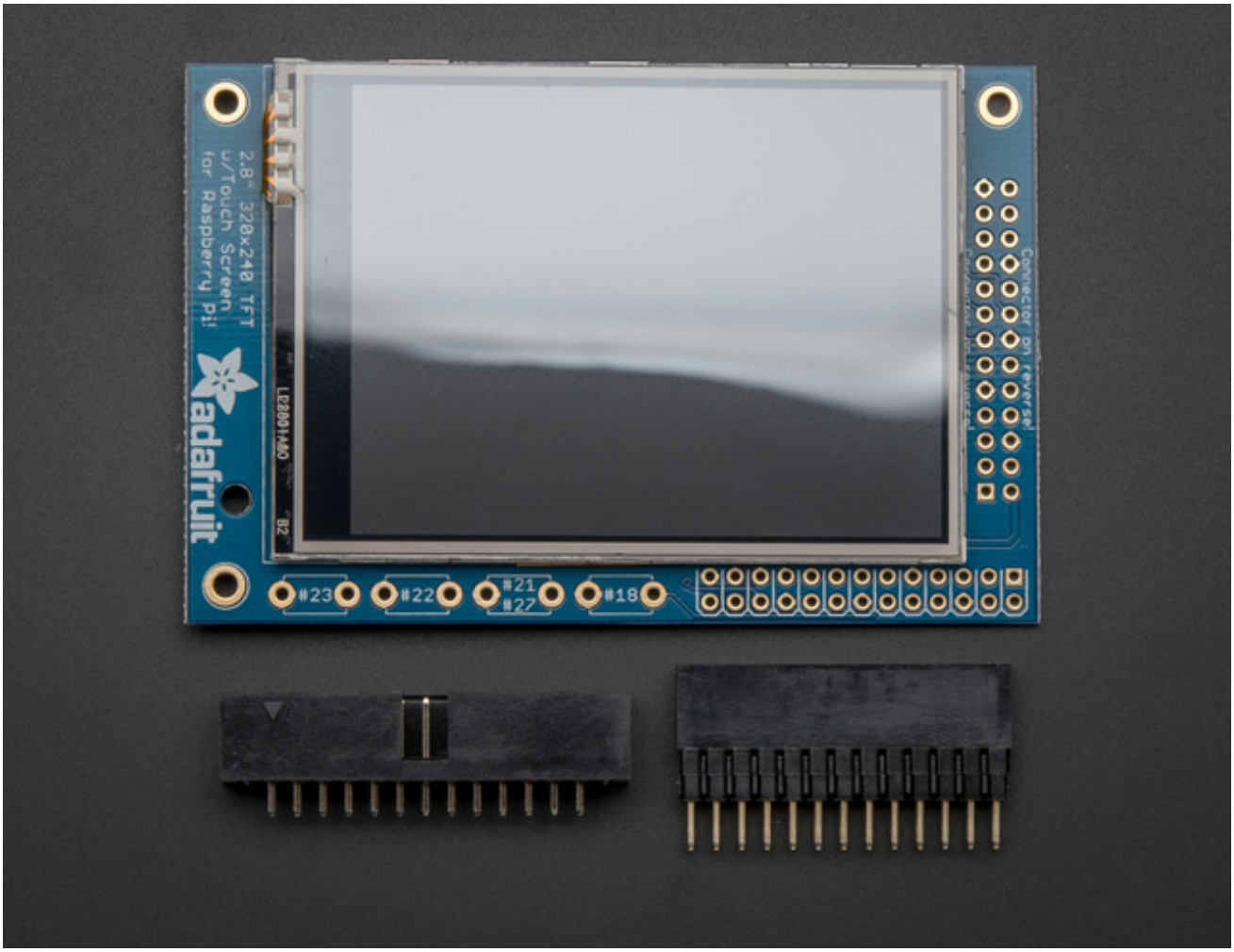
To make it super easy for use: we've created a custom kernel package based of off Notro's awesome framebuffer work, so you can install it over your existing Raspbian (or derivative) images in just a few commands.


- Try a different -vo driver (-vo help for a list)
or try -framedrop!
Big Buck
BUNNY
V: 28.0 V: 27.1 A-V: 0.875 ct: 2.681 0/ 0 64%
adafruit
Connector on reverse!

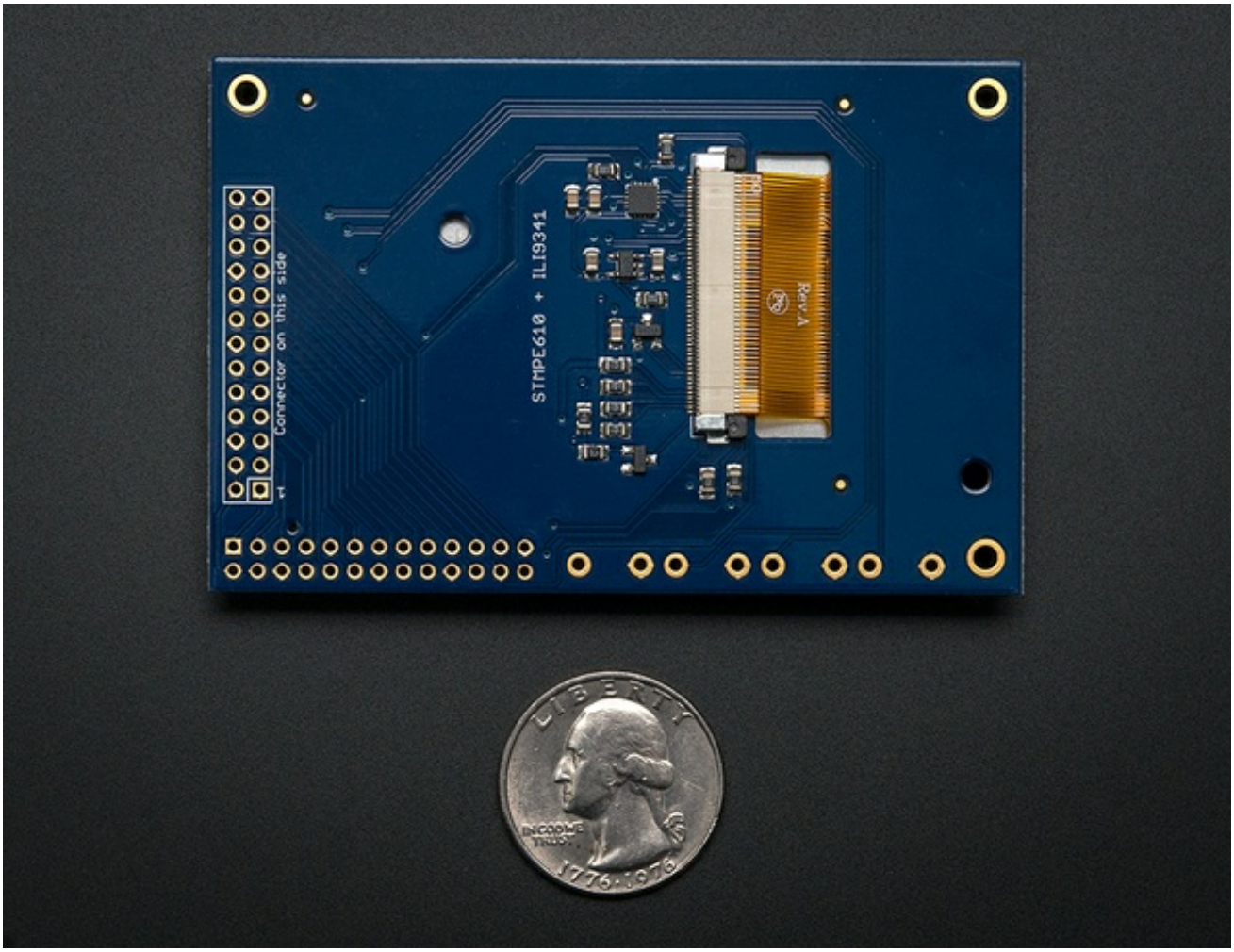
This tutorial series shows you how to install the software, as well as calibrate the touchscreen, splay videos, display images such as from your PiCam and more!

Assembly

This tutorial page is for PiTFT that came as a kit. If your PiTFT is already assembled, skip this step!

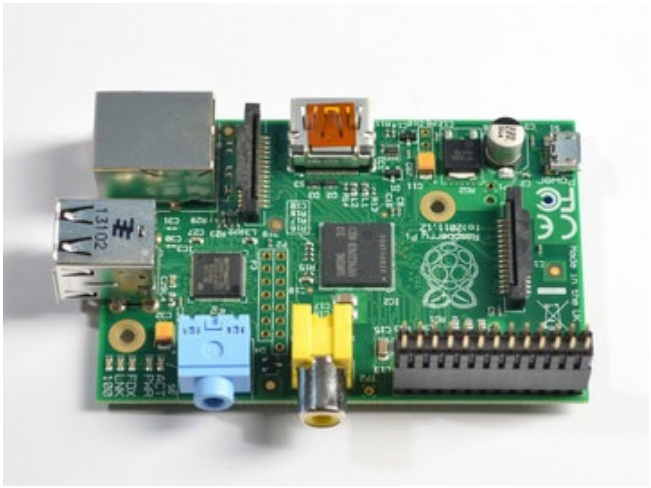


Before you start check that you have the parts you need: an assembled PiTFT plate with the 2.8" screen, extra tall female header and the 2x13 IDC socket. Note that it is normal for the screen to be 'loose' - this is so its easier for you to solder the connector on!



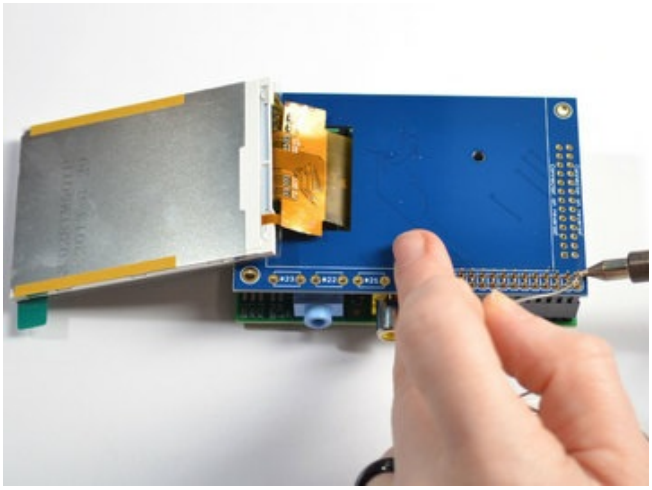
Check also on the back that the TFT is attached and that the flex connector is seated into the onboard FPC socket.

The easiest way to attach the header is if you have a Raspberry Pi as a 'stand' - make sure its powered off & unplugged!

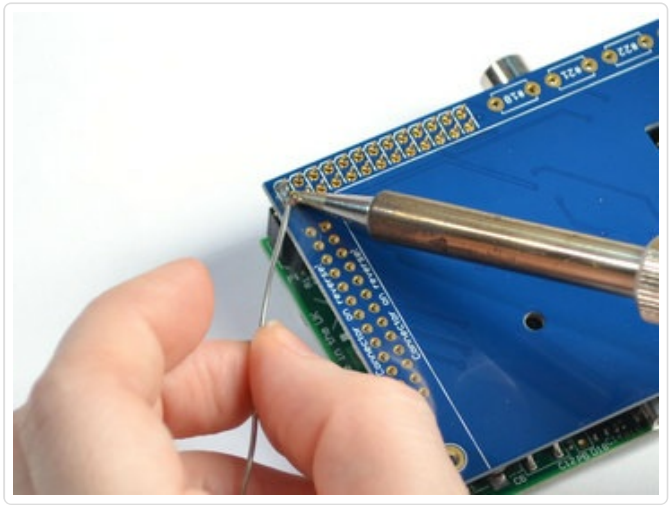


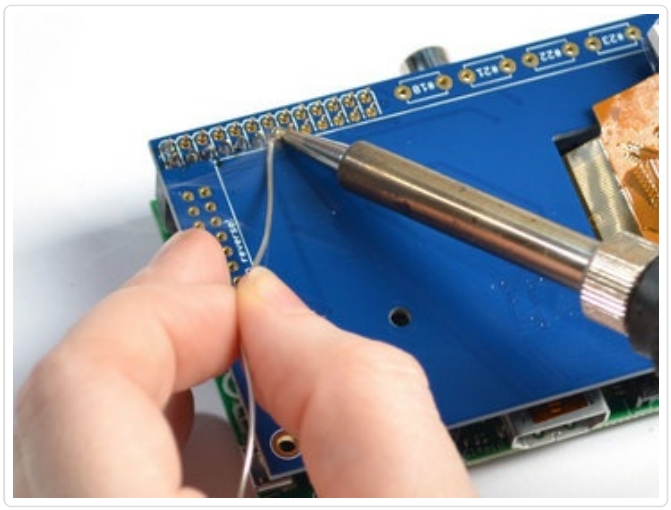
Plug the extra tall female header into the GPIO port on the Pi as shown. Make sure its seated nice and flat

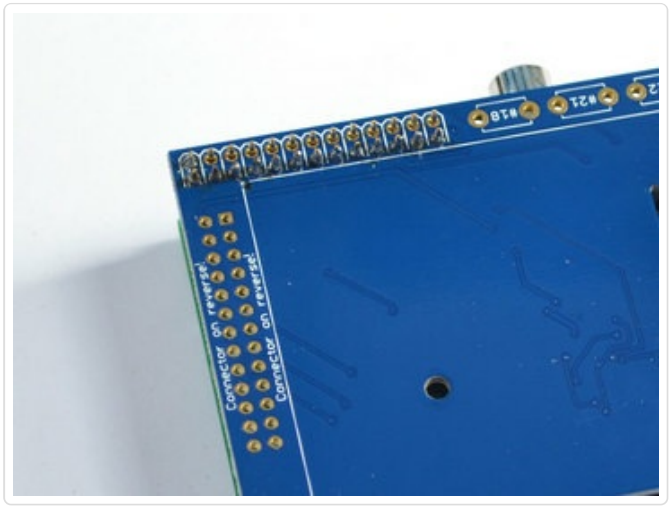
Place the PiTFT shield on top so all the pins stick through the connector on the side. Gently flip the TFT so its off to the side and wont be in your way while you solder



Heat up your soldering iron, and grab some solder. Start by tack-soldering one of the corners while pressing on the plate to make it sit flat. Once you have one or two pins done you can continue to solder each of the pins.

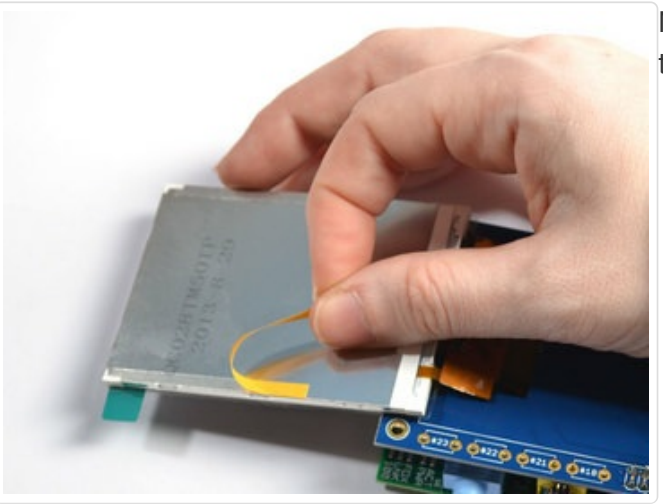






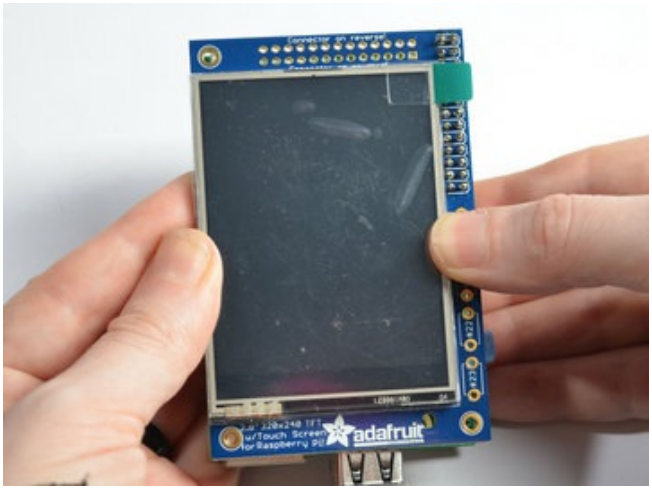
Before attaching the display, check that all the pins are soldered nicely and there's no bridging, cold

solder, shorts, or unsoldered pins.

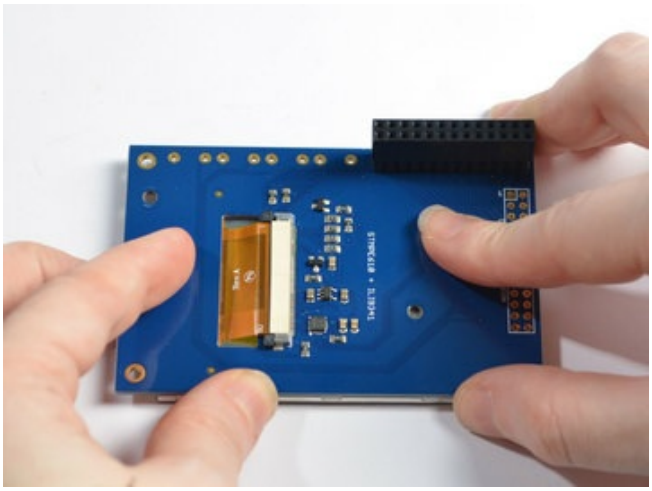


Now we can attach the screen. Remove the two thin tape cover strips.

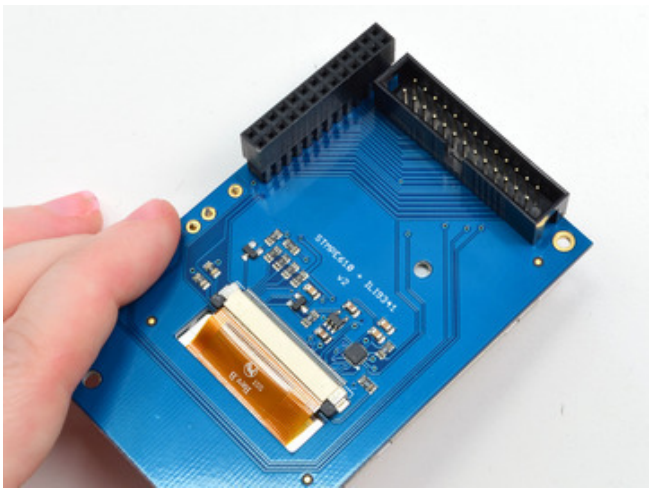
Line up the screen on the white outline, make sure there's some space from the header you just soldered in and the metal sides of the screen. As long as you don't really press down on the screen you can reposition it once or twice.



Once you have the screen so it is definitely not touching the header, you can gently press on the sides to secure the tape.

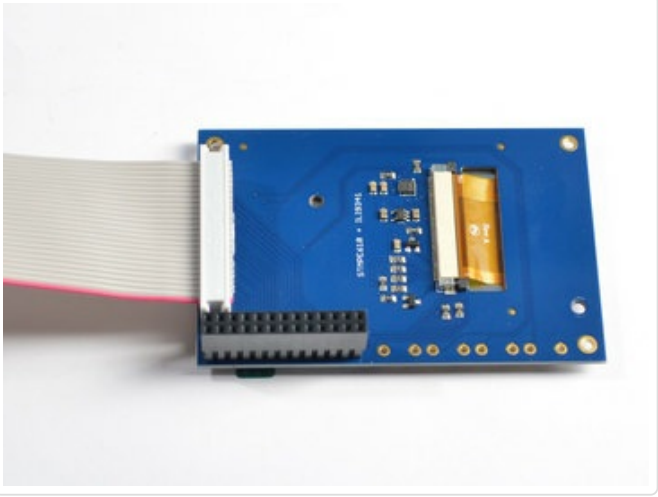


If the protective plastic cover is still on the screen you can press it against a clean table from above. That way you will really securely attach it!



If you want to attach an Adafruit Cobbler or similar, you can solder in the optional 2x13 IDC on the **bottom** of the screen as shown here. This will keep the top side clean and flat. Solder in all 26 pins

The picture shows a 2x13 male header. We've since updated this product to include an IDC socket so it's easier to add a cobbler. Both will work, though!



You can attach a 26-pin IDC cable just make sure the pin 1 indicator is on the right as indicated in this photo - there's also a #1 marking on the PCB!

Easy Install

The PiTFT requires kernel support and a couple other things to make it a nice stand-alone display. We have a detailed step-by-step setup for hackers who want to tweak, customize or understand the PiTFT setup. If you just want to get going, check out the following for easy-install instructions!

Ready to go image

If you want to start with a fresh image, we have one for Raspbian - click here to download it and install into a new SD card. Unzip and follow the classic SD card burning tutorials (http://adafru.it/aMW)

Download PiTFT 2.4" and 2.8" Resistive Image for Pi 1 and Pi 2 (March 3, 2015)

http://adafru.it/elo

This image is customized for the RESISTIVE touch 2.8" TFT, also known as PID #1601, or the Resistive 2.4" HAT, a.k.a PID #2455 Not for PID #1983

Previous images:

- Raspbian 2014/06/20-based image (http://adafru.it/dSM)
- Raspbian 2014/09/09-based image (http://adafru.it/e12)

DIY Installer script

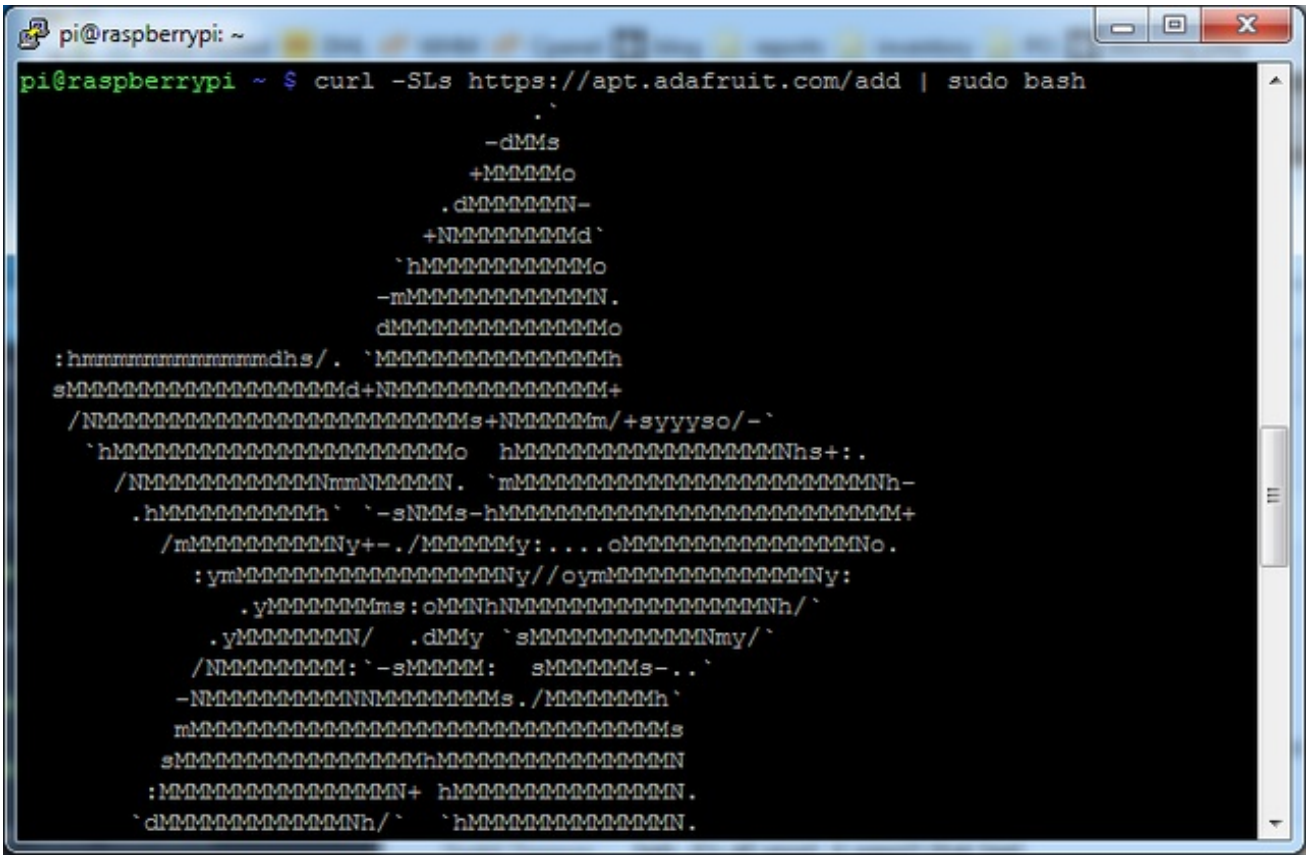
If you don't want to download an image, you can run our installation package helper from inside your existing Raspbian install. It will download the kernel add-ons, and configure your Pi for PiTFT joy

The helper is available for perusal here (http://adafru.it/eIn) if you are interested in how it works

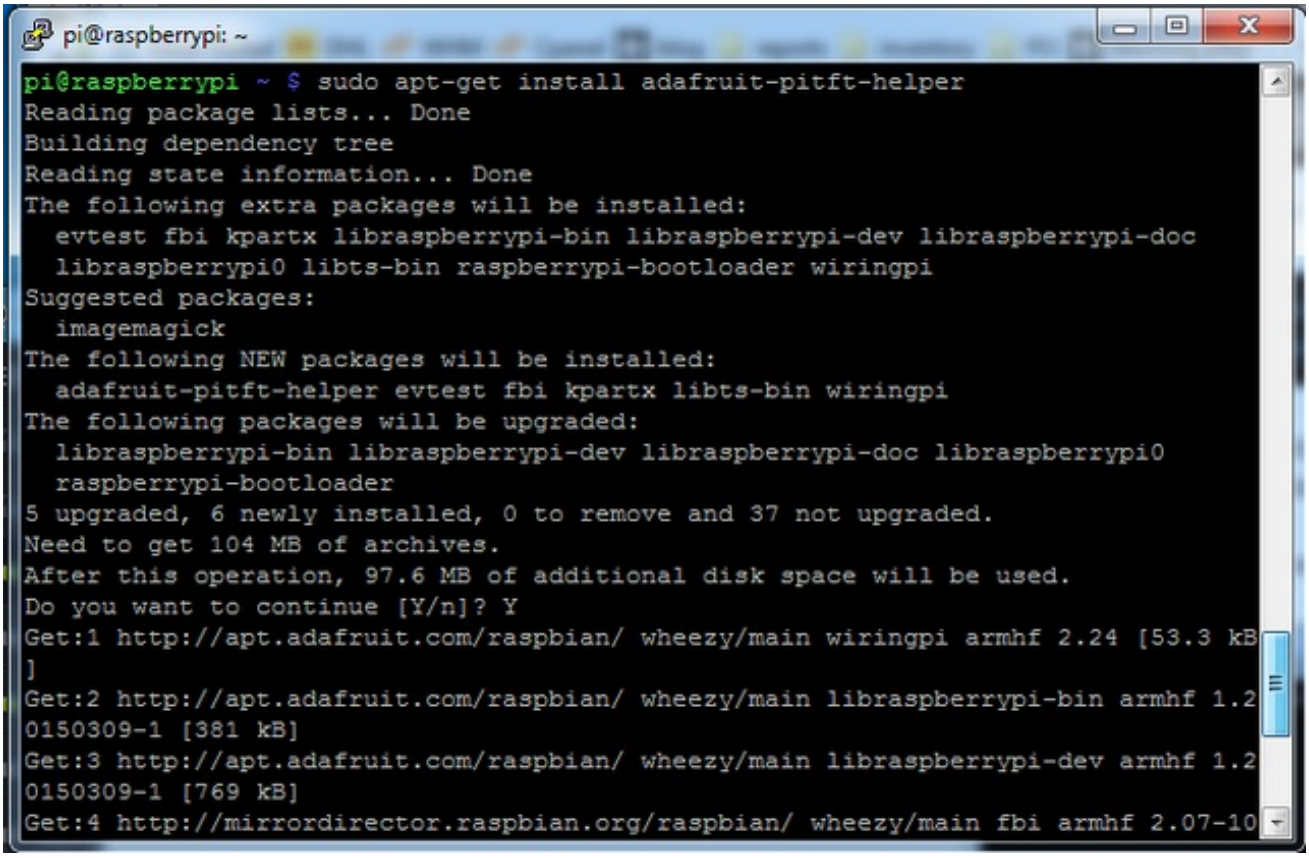
To download and run it, simply run the following commands:

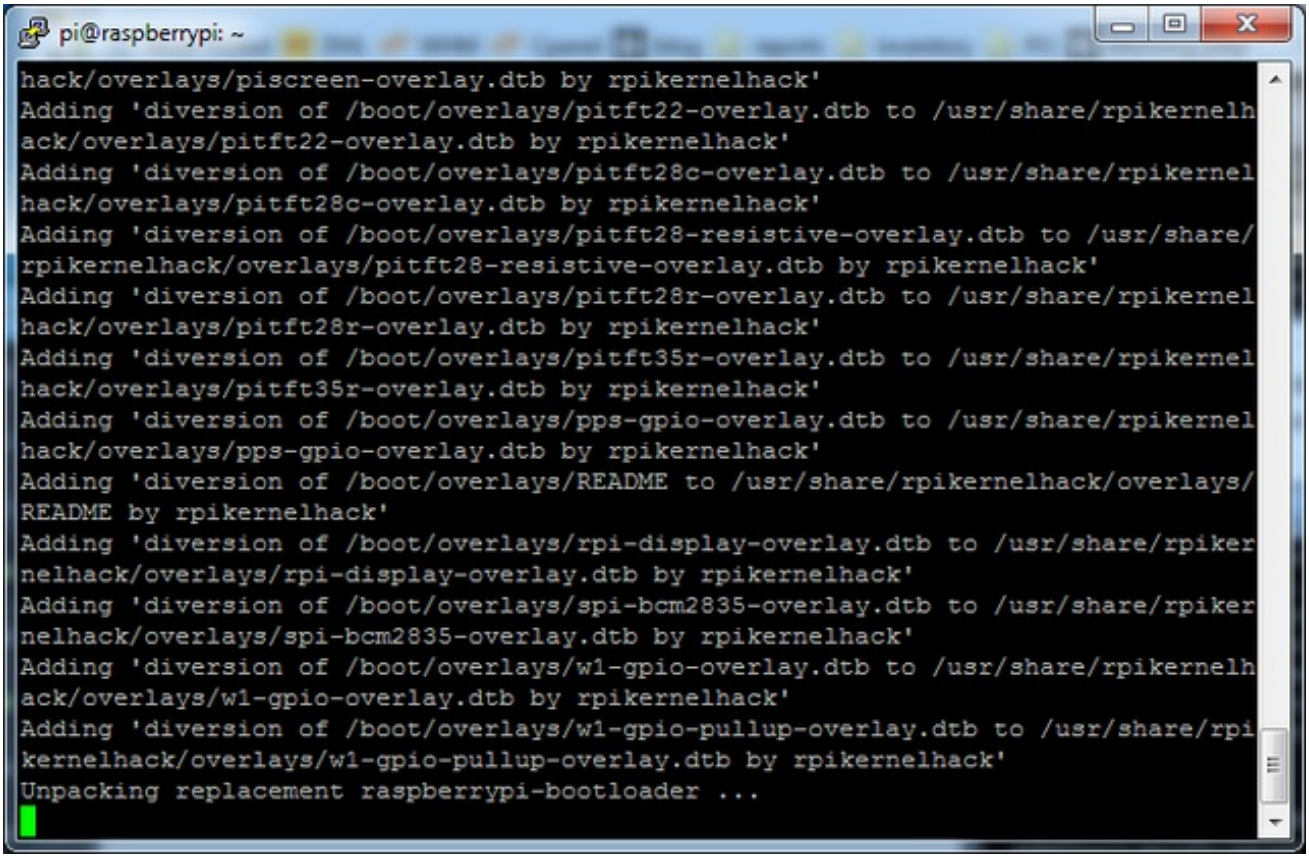
```
curl -SLs https://apt.adafruit.com/add | sudo bash
sudo apt-get install -y adafruit-pitft-helper
```

The first command adds **apt.adafruit.com** to your repository list, so you can grab code directly from adafruit's servers

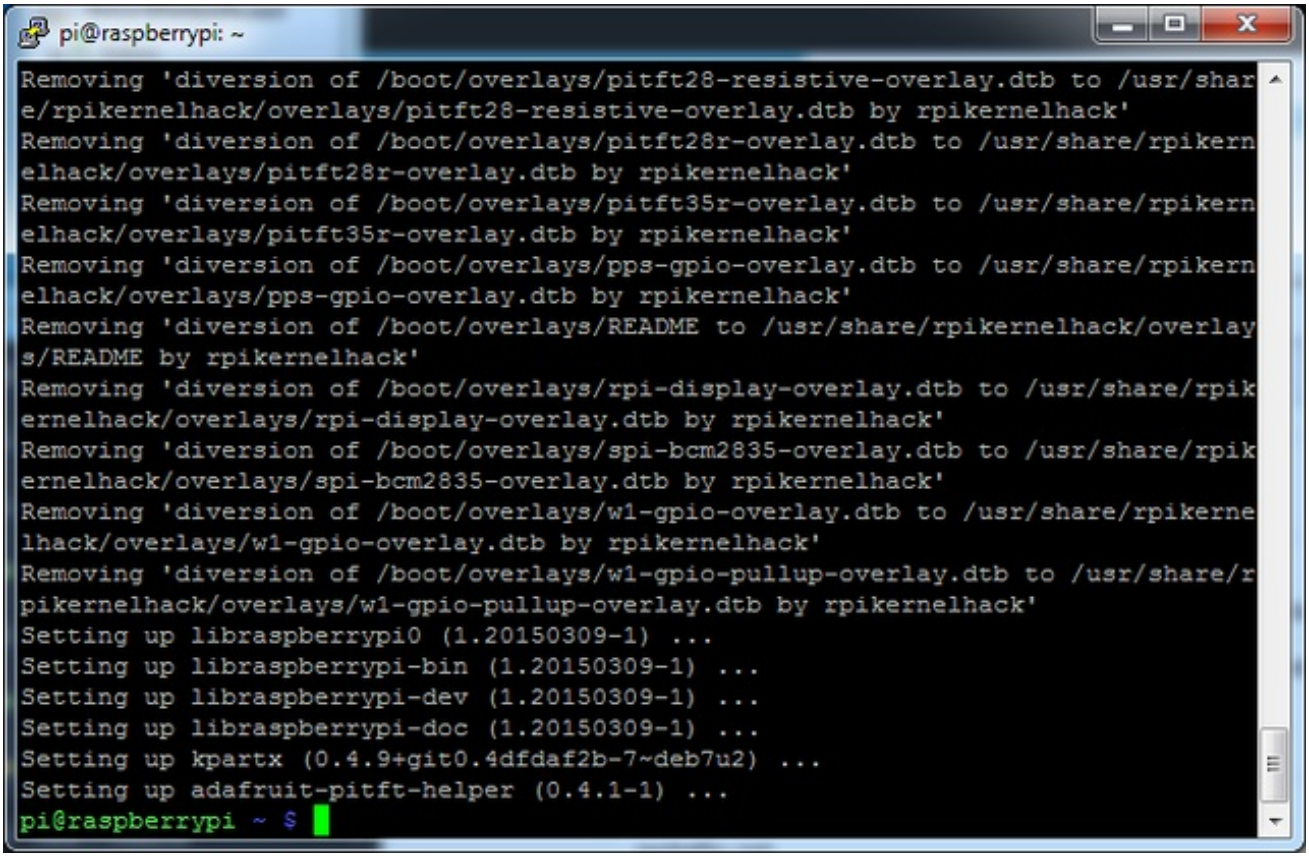


The second does the actual download and installation, it'll take a while because there's a lot of software to replace for PiTFT support


pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo apt-get install adafruit-pitft-helper
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
evtest fbi kpartx libraspberrypi-bin libraspberrypi-dev libraspberrypi-doc
libraspberrypi0 libts-bin raspberrypi-bootloader wiringpi
Suggested packages:
imagemagick
The following NEW packages will be installed:
adafruit-pitft-helper evtest fbi kpartx libts-bin wiringpi
The following packages will be upgraded:
libraspberrypi-bin libraspberrypi-dev libraspberrypi-doc libraspberrypi0
raspberrypi-bootloader
5 upgraded, 6 newly installed, 0 to remove and 37 not upgraded.
Need to get 104 MB of archives.
After this operation, 97.6 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get:1 http://apt.adafruit.com/raspbian/ wheezy/main wiringpi armhf 2.24 [53.3 kB
]
Get:2 http://apt.adafruit.com/raspbian/ wheezy/main libraspberrypi-bin armhf 1.2
0150309-1 [381 kB]
Get:3 http://apt.adafruit.com/raspbian/ wheezy/main libraspberrypi-dev armhf 1.2
0150309-1 [769 kB]
Get:4 http://mirrordirector.raspbian.org/raspbian/ wheezy/main fbi armhf 2.07-10


pi@raspberrypi: ~
hack/overlays/piscreen-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pitft22-overlay.dtb to /usr/share/rpikernelh
ack/overlays/pitft22-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pitft28c-overlay.dtb to /usr/share/rpikernel
hack/overlays/pitft28c-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pitft28-resistive-overlay.dtb to /usr/share/
rpikernelhack/overlays/pitft28-resistive-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pitft28r-overlay.dtb to /usr/share/rpikernel
hack/overlays/pitft28r-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pitft35r-overlay.dtb to /usr/share/rpikernel
hack/overlays/pitft35r-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pps-gpio-overlay.dtb to /usr/share/rpikernel
hack/overlays/pps-gpio-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/README to /usr/share/rpikernelhack/overlays/
README by rpikernelhack'
Adding 'diversion of /boot/overlays/rpi-display-overlay.dtb to /usr/share/rpiker
nelhack/overlays/rpi-display-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/spi-bcm2835-overlay.dtb to /usr/share/rpiker
nelhack/overlays/spi-bcm2835-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/w1-gpio-overlay.dtb to /usr/share/rpikernelh
ack/overlays/w1-gpio-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/w1-gpio-pullup-overlay.dtb to /usr/share/rpi
kernelhack/overlays/w1-gpio-pullup-overlay.dtb by rpikernelhack'
Unpacking replacement raspberrypi-bootloader ...

It's normal for the Pi to pause at this step for up to 20 minutes, theres a lot of kernel software to replace

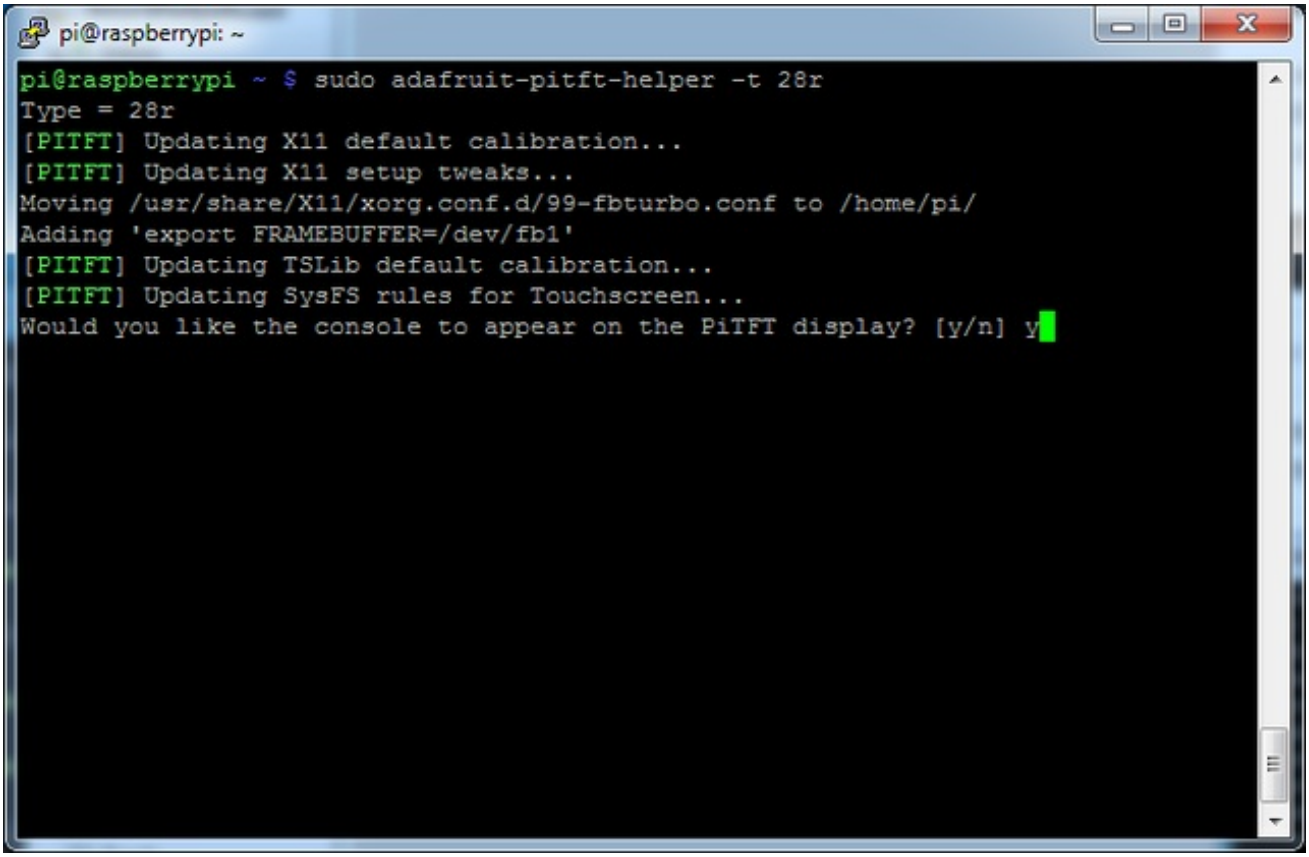

pi@raspberrypi: ~
Removing 'diversion of /boot/overlays/pitft28-resistive-overlay.dtb to /usr/share/rpikernelhack/overlays/pitft28-resistive-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/pitft28r-overlay.dtb to /usr/share/rpikernelhack/overlays/pitft28r-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/pitft35r-overlay.dtb to /usr/share/rpikernelhack/overlays/pitft35r-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/pps-gpio-overlay.dtb to /usr/share/rpikernelhack/overlays/pps-gpio-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/README to /usr/share/rpikernelhack/overlays/README by rpikernelhack'
Removing 'diversion of /boot/overlays/rpi-display-overlay.dtb to /usr/share/rpikernelhack/overlays/rpi-display-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/spi-bcm2835-overlay.dtb to /usr/share/rpikernelhack/overlays/spi-bcm2835-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/w1-gpio-overlay.dtb to /usr/share/rpikernelhack/overlays/w1-gpio-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/w1-gpio-pullup-overlay.dtb to /usr/share/rpikernelhack/overlays/w1-gpio-pullup-overlay.dtb by rpikernelhack'
Setting up libraspberrypi0 (1.20150309-1) ...
Setting up libraspberrypi-bin (1.20150309-1) ...
Setting up libraspberrypi-dev (1.20150309-1) ...
Setting up libraspberrypi-doc (1.20150309-1) ...
Setting up kpartx (0.4.9+git0.4dfdaf2b-7~deb7u2) ...
Setting up adafruit-pitft-helper (0.4.1-1) ...
pi@raspberrypi ~ $

OK now the kernel is installed, all you have to do is run the helper which will configure the kernel device tree overlays and add the few configurations to make the console show up, etc.

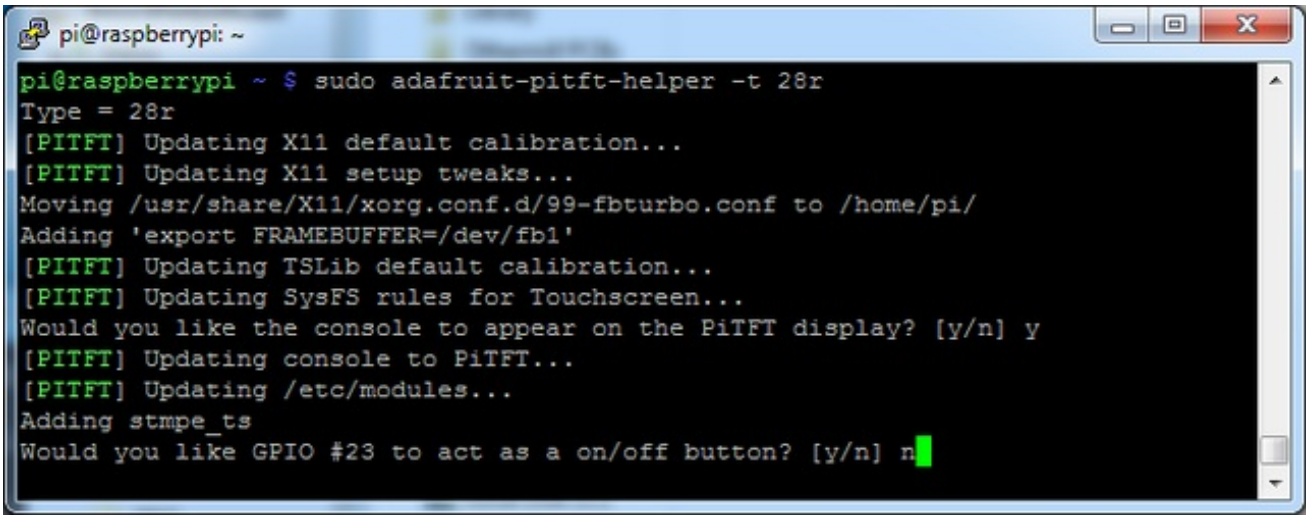
```
sudo adafruit-pitft-helper -t 28r
```

This will install the "2.8 Resistive" type of PiTFT into the current install. This is the same as the 2.4" Resistive screen too (same resolution, pinout, etc.)

At the end you will be prompted on whether you want the text console to appear on the PiTFT. Answer Y or N depending on your personal desires!


pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo adafruit-pitft-helper -t 28r
Type = 28r
[PITFT] Updating X11 default calibration...
[PITFT] Updating X11 setup tweaks...
Moving /usr/share/X11/xorg.conf.d/99-fbturbo.conf to /home/pi/
Adding 'export FRAMEBUFFER=/dev/fb1'
[PITFT] Updating TSLib default calibration...
[PITFT] Updating SysFS rules for Touchscreen...
Would you like the console to appear on the PiTFT display? [y/n] y

You will also be prompted on whether you want one of the tactile buttons to act as an 'on off' switch. Answer Y or N depending on your personal desires!


pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo adafruit-pitft-helper -t 28r
Type = 28r
[PITFT] Updating X11 default calibration...
[PITFT] Updating X11 setup tweaks...
Moving /usr/share/X11/xorg.conf.d/99-fbturbo.conf to /home/pi/
Adding 'export FRAMEBUFFER=/dev/fb1'
[PITFT] Updating TSLib default calibration...
[PITFT] Updating SysFS rules for Touchscreen...
Would you like the console to appear on the PiTFT display? [y/n] y
[PITFT] Updating console to PiTFT...
[PITFT] Updating /etc/modules...
Adding stmpe_ts
Would you like GPIO #23 to act as a on/off button? [y/n] n

Thats it!

Run **sudo reboot** to try out your fancy new PiTFT :)

Detailed Installation

If you've grabbed our Easy Install image, or use the script, this step is not required, it's already done! This is just for advanced users who are curious on how to configure and customize the kernel install

In the next few steps we'll cover the **detailed** installation procedure. Chances are, you should grab the Easy Install image or script. If you have some interest in the details of how we install the PiTFT setup, read on!

In order to add support for the 2.4" or 2.8" TFT and touchscreen, we'll need to install a new Linux Kernel. Lucky for you, we created a kernel package that you can simply install *over* your current Raspbian (or Raspbian-derived) install instead of needing a whole new image. This makes it easier to keep your install up-to-date.

To use our kernel .deb files you must be using Raspbian or derivative. This wont work with Arch or other Linux flavors. As Raspbian is the official OS for the Pi, that's the only Linux we will support!

Others can recompile their own kernel using our patchfile (http://adafru.it/cY2), but we have no tutorial or support or plans for such.

Before you start

You'll need a working install of Raspbian with network access. If you need help getting that far, check out our collection of Pi tutorials (http://adafru.it/aWq).

We'll be doing this from a console cable connection, but you can just as easily do it from the direct HDMI/TV console or by SSH'ing in. Whatever gets you to a shell will work!

Also, run **sudo apt-get update** !

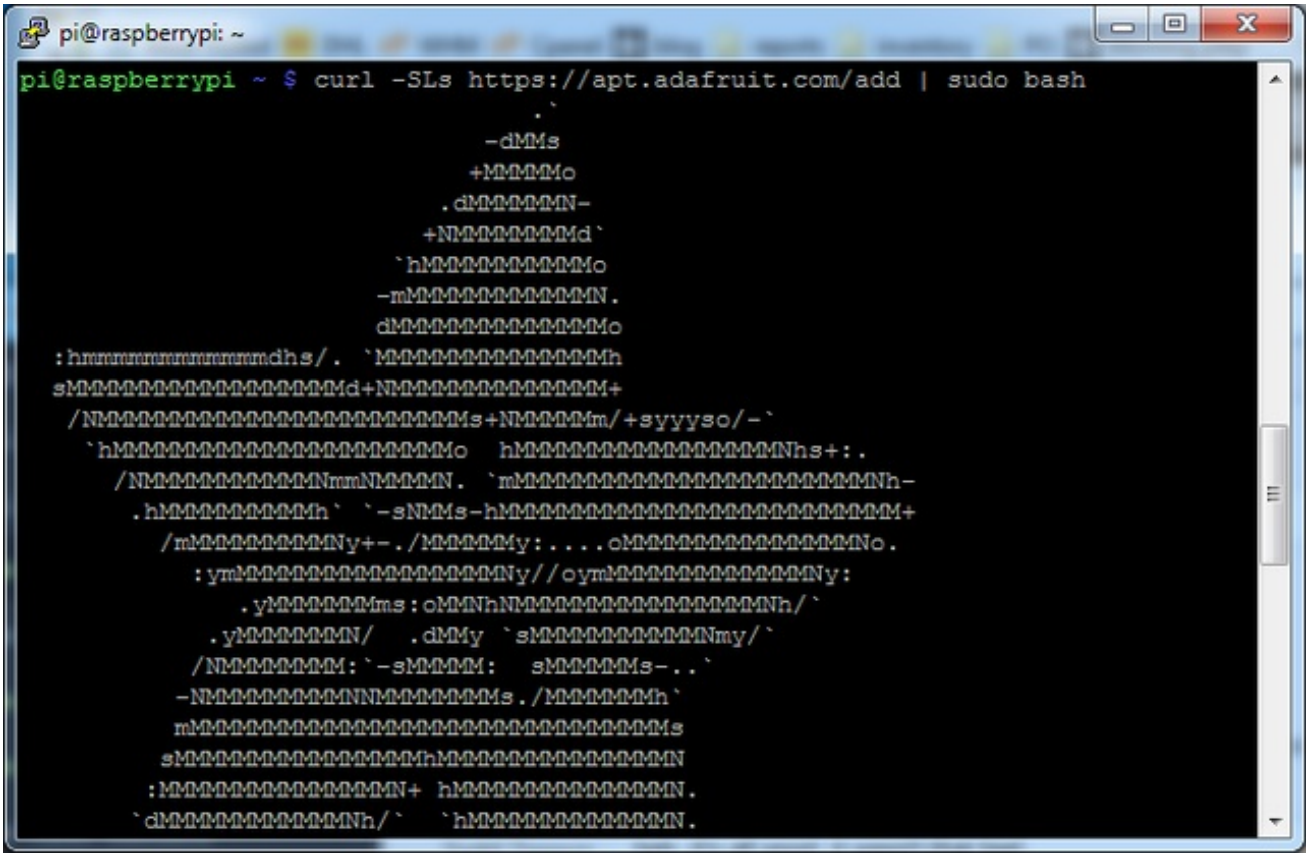
> To run these all the setup and config commands you'll need to be logged into a proper Terminal - use ssh, a console cable, or the main text console (on a TV). The WebIDE console may not work.

Download & Install Kernel

The only way we're distributing the PiTFT kernel packages right now is thru apt.adafruit.com so you'll still need to run:

```
curl -SLs https://apt.adafruit.com/add | sudo bash
```

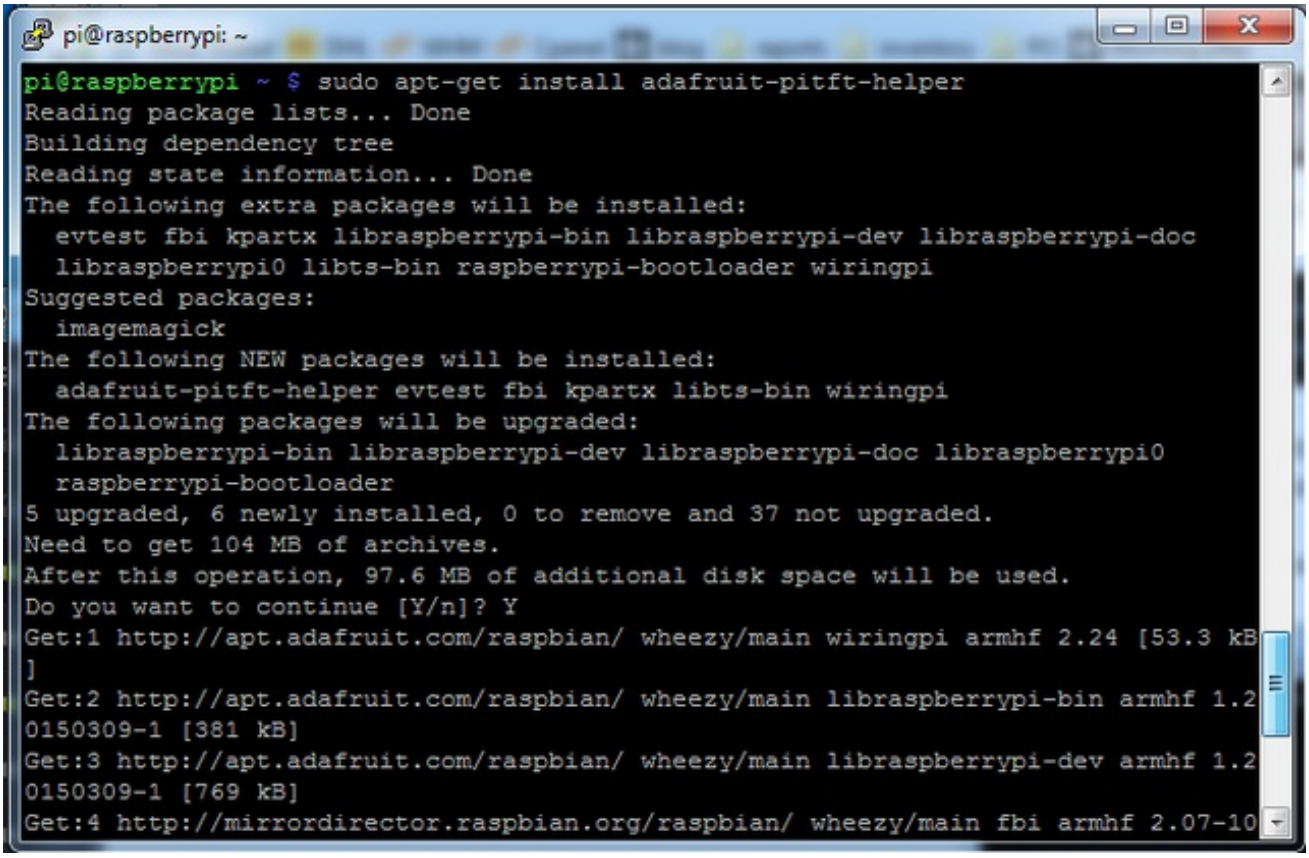
To add apt.adafruit.com to your list of software sources

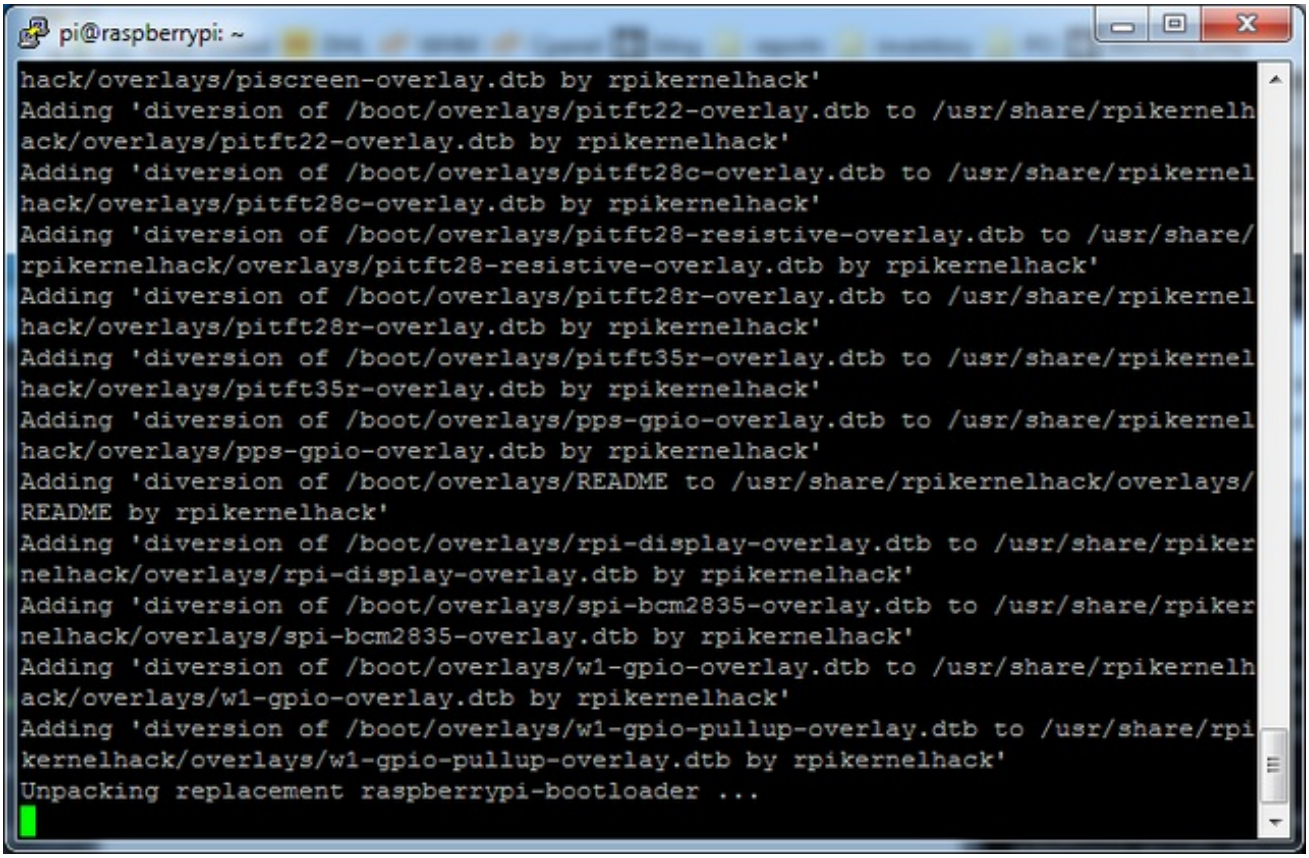


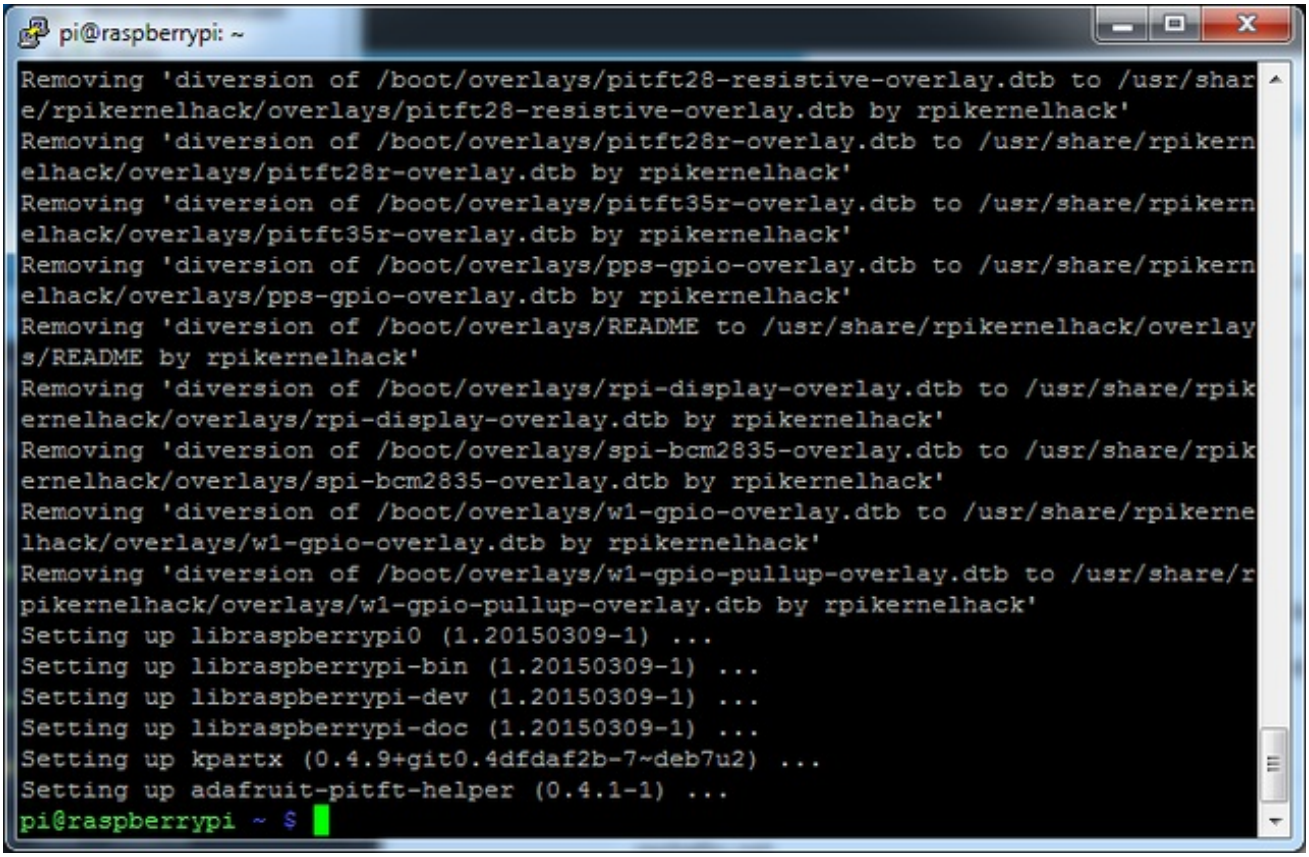
Then install the kernel with

```
sudo apt-get install -y adafruit-pitft-helper
```

This will take a up to 20 minutes so go make a sandwich or coffee. It takes longer than it used to because there's now 2 kernels (v6 and v7 arm) and 2 kernel module directories.


pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo apt-get install adafruit-pitft-helper
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
evtest fbi kpartx libraspberrypi-bin libraspberrypi-dev libraspberrypi-doc
libraspberrypi0 libts-bin raspberrypi-bootloader wiringpi
Suggested packages:
imagemagick
The following NEW packages will be installed:
adafruit-pitft-helper evtest fbi kpartx libts-bin wiringpi
The following packages will be upgraded:
libraspberrypi-bin libraspberrypi-dev libraspberrypi-doc libraspberrypi0
raspberrypi-bootloader
5 upgraded, 6 newly installed, 0 to remove and 37 not upgraded.
Need to get 104 MB of archives.
After this operation, 97.6 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get:1 http://apt.adafruit.com/raspbian/ wheezy/main wiringpi armhf 2.24 [53.3 kB
]
Get:2 http://apt.adafruit.com/raspbian/ wheezy/main libraspberrypi-bin armhf 1.2
0150309-1 [381 kB]
Get:3 http://apt.adafruit.com/raspbian/ wheezy/main libraspberrypi-dev armhf 1.2
0150309-1 [769 kB]
Get:4 http://mirrordirector.raspbian.org/raspbian/ wheezy/main fbi armhf 2.07-10


pi@raspberrypi: ~
hack/overlays/piscreen-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pitft22-overlay.dtb to /usr/share/rpikernelh
ack/overlays/pitft22-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pitft28c-overlay.dtb to /usr/share/rpikernel
hack/overlays/pitft28c-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pitft28-resistive-overlay.dtb to /usr/share/
rpikernelhack/overlays/pitft28-resistive-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pitft28r-overlay.dtb to /usr/share/rpikernel
hack/overlays/pitft28r-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pitft35r-overlay.dtb to /usr/share/rpikernel
hack/overlays/pitft35r-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/pps-gpio-overlay.dtb to /usr/share/rpikernel
hack/overlays/pps-gpio-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/README to /usr/share/rpikernelhack/overlays/
README by rpikernelhack'
Adding 'diversion of /boot/overlays/rpi-display-overlay.dtb to /usr/share/rpiker
nelhack/overlays/rpi-display-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/spi-bcm2835-overlay.dtb to /usr/share/rpiker
nelhack/overlays/spi-bcm2835-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/w1-gpio-overlay.dtb to /usr/share/rpikernelh
ack/overlays/w1-gpio-overlay.dtb by rpikernelhack'
Adding 'diversion of /boot/overlays/w1-gpio-pullup-overlay.dtb to /usr/share/rpi
kernelhack/overlays/w1-gpio-pullup-overlay.dtb by rpikernelhack'
Unpacking replacement raspberrypi-bootloader ...

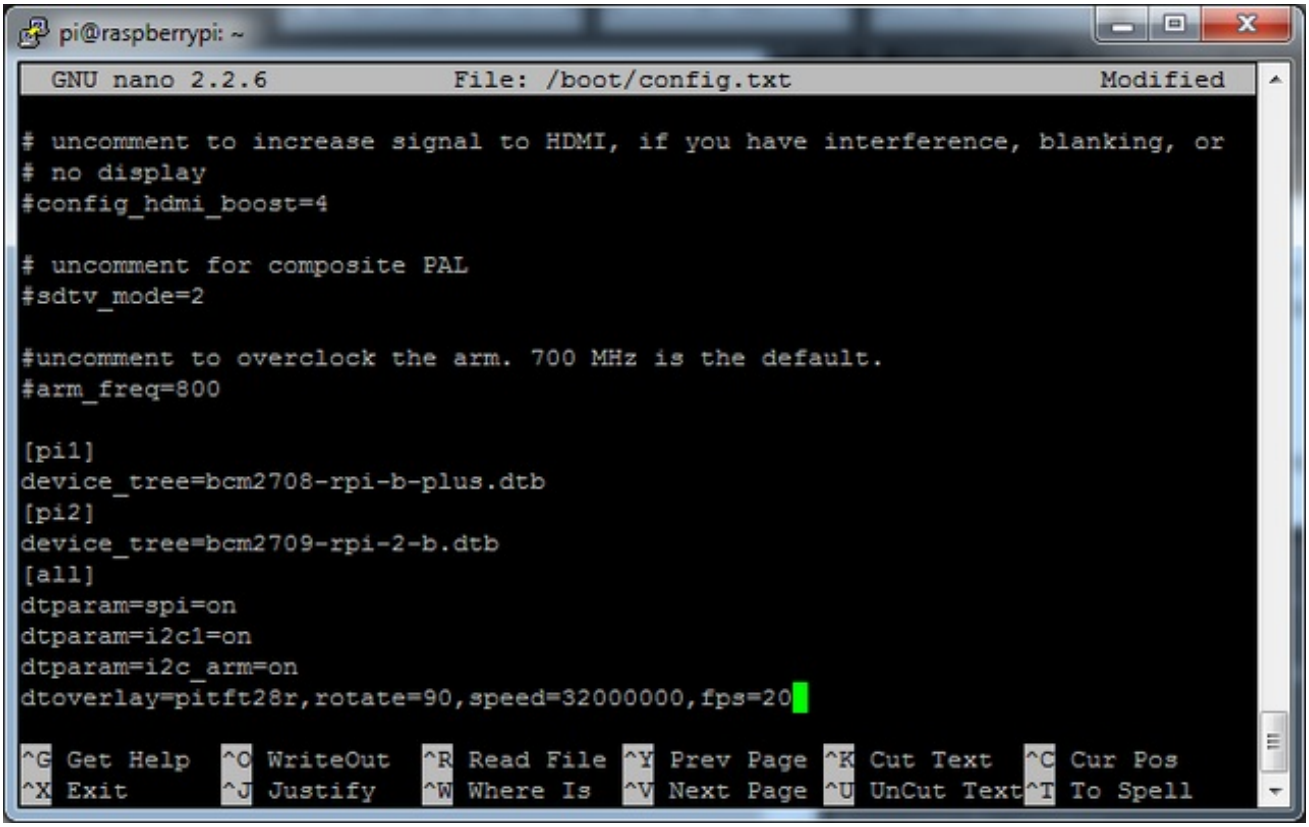

pi@raspberrypi: ~
Removing 'diversion of /boot/overlays/pitft28-resistive-overlay.dtb to /usr/share/rpikernelhack/overlays/pitft28-resistive-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/pitft28r-overlay.dtb to /usr/share/rpikernelhack/overlays/pitft28r-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/pitft35r-overlay.dtb to /usr/share/rpikernelhack/overlays/pitft35r-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/pps-gpio-overlay.dtb to /usr/share/rpikernelhack/overlays/pps-gpio-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/README to /usr/share/rpikernelhack/overlays/README by rpikernelhack'
Removing 'diversion of /boot/overlays/rpi-display-overlay.dtb to /usr/share/rpikernelhack/overlays/rpi-display-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/spi-bcm2835-overlay.dtb to /usr/share/rpikernelhack/overlays/spi-bcm2835-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/w1-gpio-overlay.dtb to /usr/share/rpikernelhack/overlays/w1-gpio-overlay.dtb by rpikernelhack'
Removing 'diversion of /boot/overlays/w1-gpio-pullup-overlay.dtb to /usr/share/rpikernelhack/overlays/w1-gpio-pullup-overlay.dtb by rpikernelhack'
Setting up libraspberrypi0 (1.20150309-1) ...
Setting up libraspberrypi-bin (1.20150309-1) ...
Setting up libraspberrypi-dev (1.20150309-1) ...
Setting up libraspberrypi-doc (1.20150309-1) ...
Setting up kpartx (0.4.9+git0.4dfdaf2b-7~deb7u2) ...
Setting up adafruit-pitft-helper (0.4.1-1) ...
pi@raspberrypi ~ $

OK since you're not going to run the helper, lets add the device tree overlay manually. Edit /boot/config.txt with

sudo nano /boot/config.txt

and add the following lines at the end:

```
[pi1]
device_tree=bcm2708-rpi-b-plus.dtb
[pi2]
device_tree=bcm2709-rpi-2-b.dtb
[all]
dtparam=spi=on
dtparam=i2c1=on
dtparam=i2c_arm=on
dtoverlay=pitft28r,rotate=90,speed=32000000,fps=20
```


pi@raspberrypi: ~
GNU nano 2.2.6 File: /boot/config.txt Modified
uncomment to increase signal to HDMI, if you have interference, blanking, or
no display
#config_hdmi_boost=4
uncomment for composite PAL
#sdtv_mode=2
#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800
[pi1]
device_tree=bcm2708-rpi-b-plus.dtb
[pi2]
device_tree=bcm2709-rpi-2-b.dtb
[all]
dtparam=spi=on
dtparam=i2c1=on
dtparam=i2c_arm=on
dtoverlay=pitft28r,rotate=90,speed=32000000,fps=20
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

The **rotate=** variable tells the driver to rotate the screen **0 90 180** or **270** degrees.
0 is portrait, with the bottom near theUSB jacks
90 is landscape, with the bottom of the screen near the headphone jack
180 is portrait, with the top near the USB jacks
270 is landscape, with the top of the screen near the headphone jack.
You can change this file with **nano** and reboot to make the change stick.

The **speed=** variable tells the driver how to fast to drive the display. 32MHz (**32000000**) is a good place to start but if your screen is acting funny, try taking it down to 16MHz (**16000000**) *especially* if you're doing something like using a GPIO extender to put the screen away from the Pi.

Save the file. Now we'll just reboot to let it all sink in.

> **sudo shutdown -h now** (if you don't have the TFT installed, shutdown, place the TFT on the Pi and re-power)

or

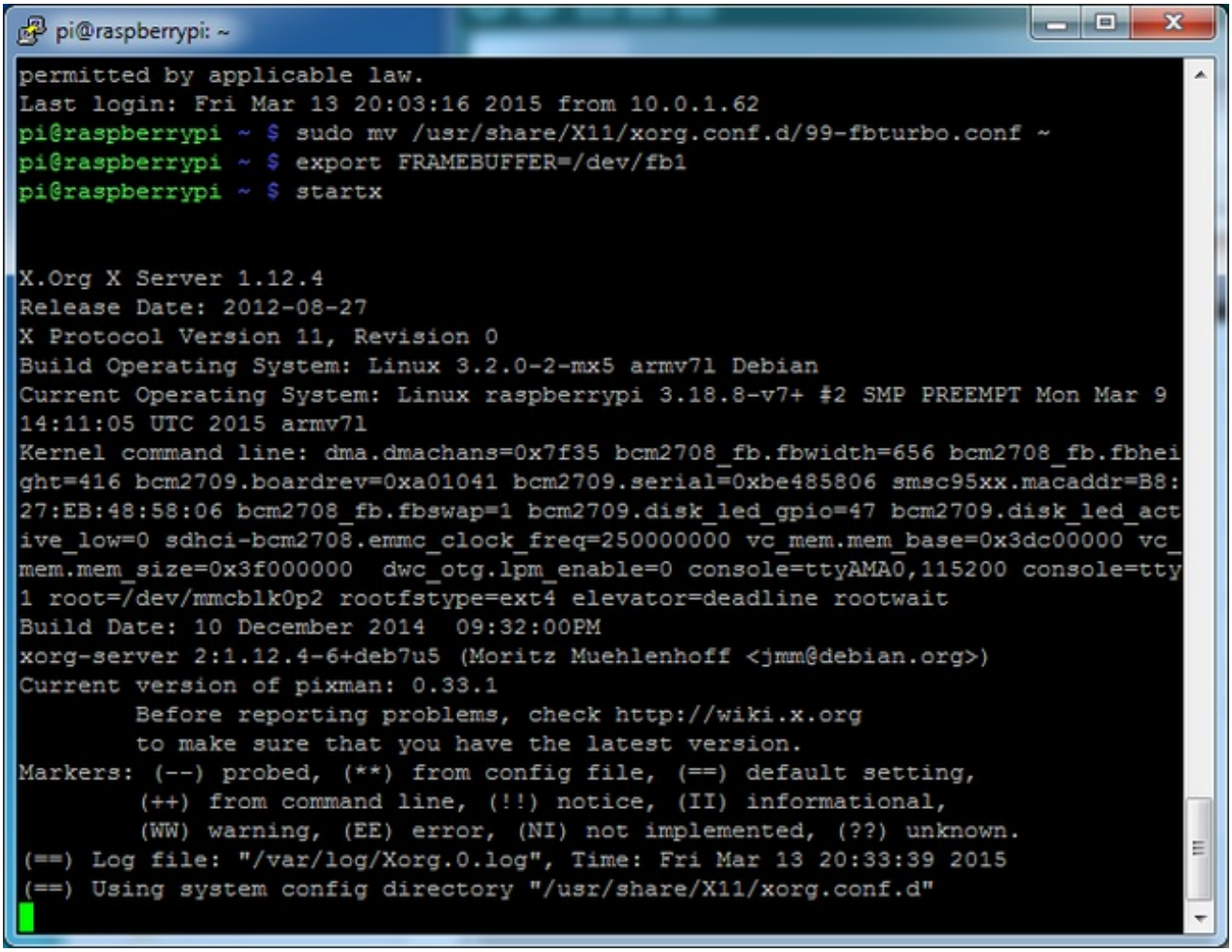
> **sudo reboot** (if you have the TFT plate installed already)

When the Pi restarts, the attached PiTFT should start out all white and then turn black. That means the kernel found the display and cleared the screen. If the screen did not turn black, that means that likely there's something up with your connection or kernel install. Solder anything that needs resoldering!

Now that you're rebooted, log back in on the console/TV/SSH. There's nothing displayed on the screen yet, we'll do a test to make sure everything is perfect first!

Run the following commands to startx on the /**dev/fb1** framebuffer, a.k.a PiTFT screen:

```
sudo mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf ~
export FRAMEBUFFER=/dev/fb1
startx
```


pi@raspberrypi: ~
permitted by applicable law.
Last login: Fri Mar 13 20:03:16 2015 from 10.0.1.62
pi@raspberrypi ~ $ sudo mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf ~
pi@raspberrypi ~ $ export FRAMEBUFFER=/dev/fb1
pi@raspberrypi ~ $ startx
X.Org X Server 1.12.4
Release Date: 2012-08-27
X Protocol Version 11, Revision 0
Build Operating System: Linux 3.2.0-2-mx5 armv7l Debian
Current Operating System: Linux raspberrypi 3.18.8-v7+ #2 SMP PREEMPT Mon Mar 9
14:11:05 UTC 2015 armv7l
Kernel command line: dma.dmachans=0x7f35 bcm2708_fb.fbwidth=656 bcm2708_fb.fbhei
ght=416 bcm2709.boardrev=0xa01041 bcm2709.serial=0xbe485806 smsc95xx.macaddr=B8:
27:EB:48:58:06 bcm2708_fb.fbswap=1 bcm2709.disk_led_gpio=47 bcm2709.disk_led_act
ive_low=0 sdhci-bcm2708.emmc_clock_freq=250000000 vc_mem.mem_base=0x3dc00000 vc_
mem.mem_size=0x3f000000 dwc_otg.lpm_enable=0 console=ttyAMA0,115200 console=tty
1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
Build Date: 10 December 2014 09:32:00PM
xorg-server 2:1.12.4-6+deb7u5 (Moritz Muehlenhoff <jmm@debian.org>)
Current version of pixman: 0.33.1
Before reporting problems, check http://wiki.x.org
to make sure that you have the latest version.
Markers: (--) probed, (**) from config file, (==) default setting,
(++) from command line, (!!) notice, (II) informational,
(WW) warning, (EE) error, (NI) not implemented, (??) unknown.
(==) Log file: "/var/log/Xorg.0.log", Time: Fri Mar 13 20:33:39 2015
(==) Using system config directory "/usr/share/X11/xorg.conf.d"

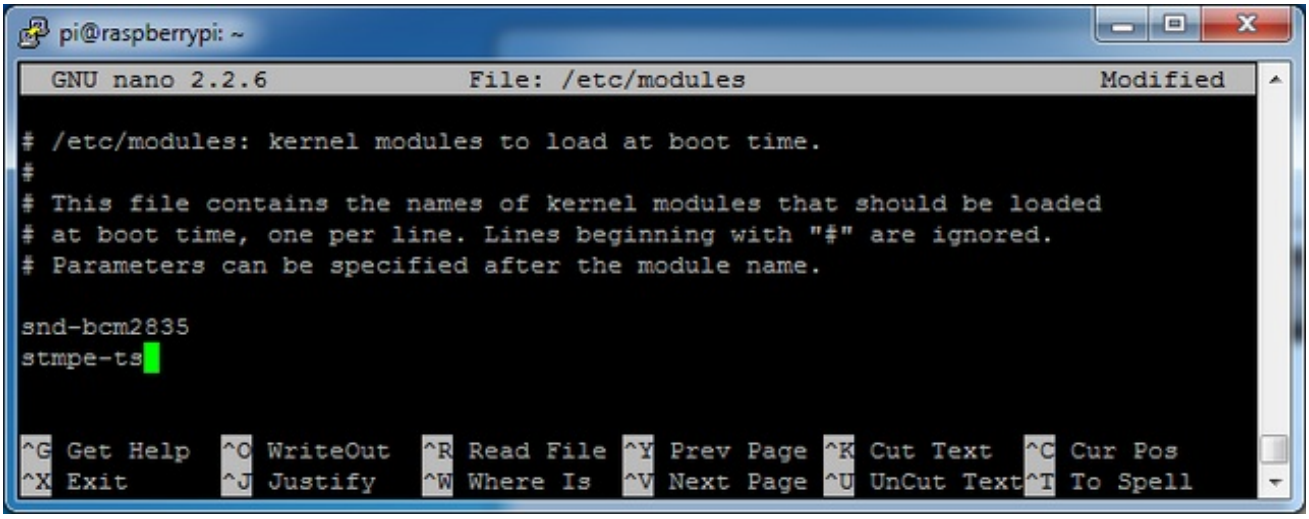
You should see the Pi desktop show up on the TFT! Congrats, you've completed the first test perfectly.

Hit Control-C in the console to quit the X server so we can continue configuration

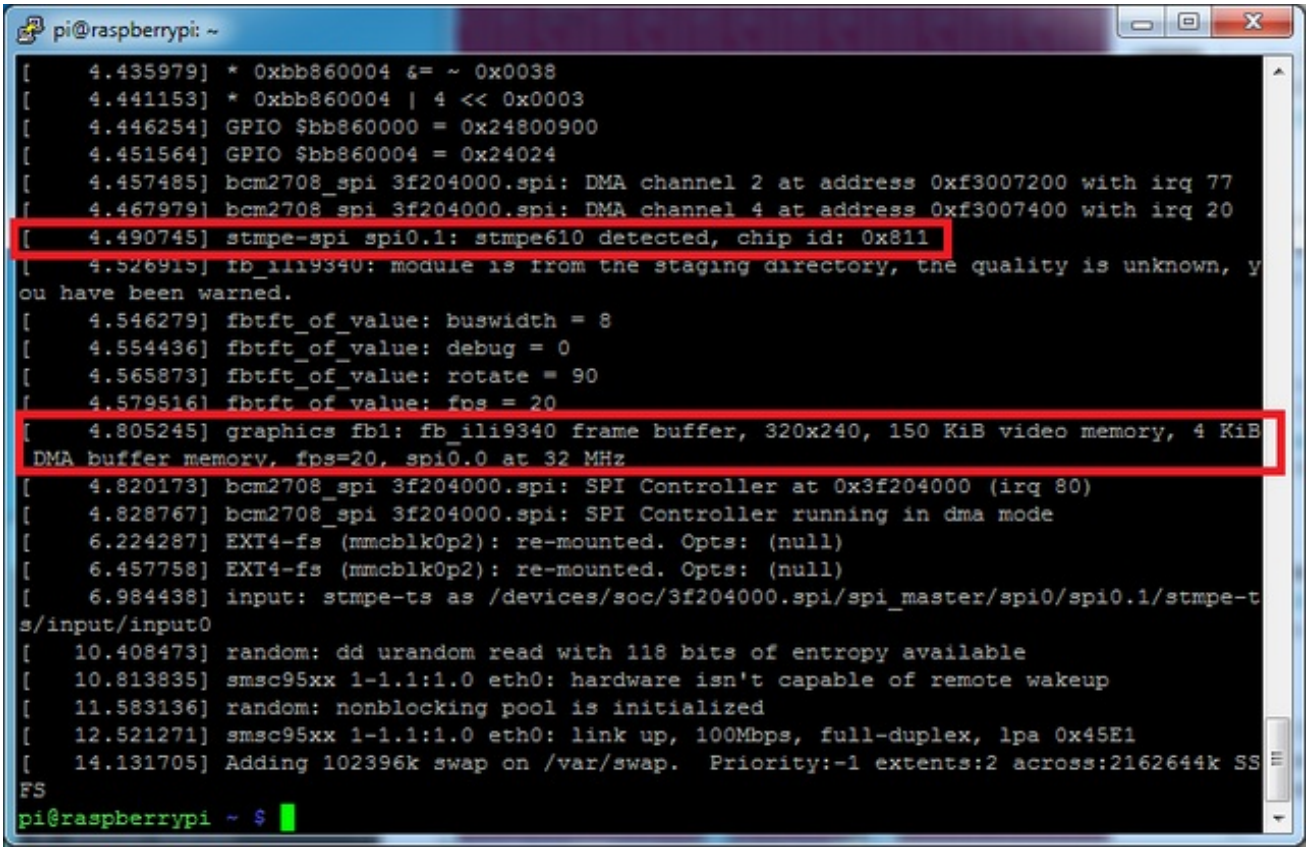
Next up we'll add support for the touch screen automatically on boot. Edit the module list with

sudo nano /etc/modules

and add **stmpe-ts** on a line at the end


pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/modules Modified
/etc/modules: kernel modules to load at boot time.
#
This file contains the names of kernel modules that should be loaded
at boot time, one per line. Lines beginning with "#" are ignored.
Parameters can be specified after the module name.
snd-bcm2835
stmpe-ts
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

Save the file and reboot the Pi with **sudo reboot** and look at the console output (or run **dmesg** in the console window after logging in) you will see the modules install. Look in particular for the STMPE610 detection and the ILI9340 screen frequency as highlighted here

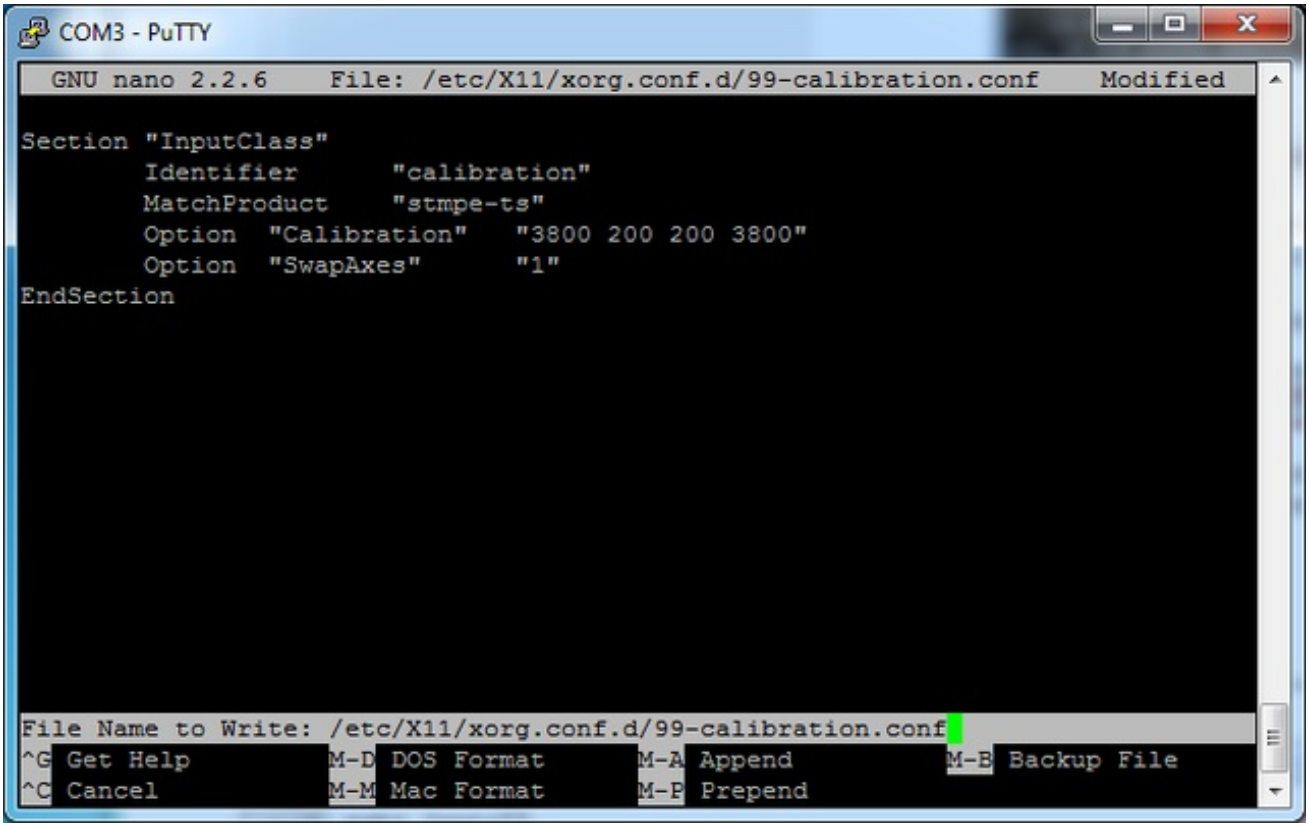

pi@raspberrypi: ~
[4.435979] * 0xbb860004 &= ~ 0x0038
[4.441153] * 0xbb860004 | 4 << 0x0003
[4.446254] GPIO $bb860000 = 0x24800900
[4.451564] GPIO $bb860004 = 0x24024
[4.457485] bcm2708_spi 3f204000.spi: DMA channel 2 at address 0xf3007200 with irq 77
[4.467979] bcm2708_spi 3f204000.spi: DMA channel 4 at address 0xf3007400 with irq 20
[4.490745] stmpe-spi spi0.1: stmpe610 detected, chip id: 0x811
[4.526915] fb_ili9340: module is from the staging directory, the quality is unknown, y
ou have been warned.
[4.546279] fbtft_of_value: buswidth = 8
[4.554436] fbtft_of_value: debug = 0
[4.565873] fbtft_of_value: rotate = 90
[4.579516] fbtft_of_value: fps = 20
[4.805245] graphics fb1: fb_ili9340 frame buffer, 320x240, 150 KiB video memory, 4 KiB
DMA buffer memory, fps=20, spi0.0 at 32 MHz
[4.820173] bcm2708_spi 3f204000.spi: SPI Controller at 0x3f204000 (irq 80)
[4.828767] bcm2708_spi 3f204000.spi: SPI Controller running in dma mode
[6.224287] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[6.457758] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
[6.984438] input: stmpe-ts as /devices/soc/3f204000.spi/spi_master/spi0/spi0.1/stmpe-t
s/input/input0
[10.408473] random: dd urandom read with 118 bits of entropy available
[10.813835] smsc95xx 1-1.1:1.0 eth0: hardware isn't capable of remote wakeup
[11.583136] random: nonblocking pool is initialized
[12.521271] smsc95xx 1-1.1:1.0 eth0: link up, 100Mbps, full-duplex, lpa 0x45E1
[14.131705] Adding 102396k swap on /var/swap. Priority:-1 extents:2 across:2162644k SS
FS
pi@raspberrypi ~ $

We can set up the touchscreen for **rotate=90** configuration by doing the following (for more delicate calibration or for other rotate=XX values, see the next section)
Create the directory and new calibration configuration file:

> **sudo mkdir /etc/X11/xorg.conf.d**
> **sudo nano /etc/X11/xorg.conf.d/99-calibration.conf**

and enter in the following lines, then save.

```
Section "InputClass"
        Identifier      "calibration"
        MatchProduct    "stmpe-ts"
        Option  "Calibration"   "3800 200 200 3800"
        Option  "SwapAxes"      "1"
EndSection
```


COM3 - PuTTY
GNU nano 2.2.6 File: /etc/X11/xorg.conf.d/99-calibration.conf Modified
Section "InputClass"
Identifier "calibration"
MatchProduct "stmpe-ts"
Option "Calibration" "3800 200 200 3800"
Option "SwapAxes" "1"
EndSection
File Name to Write: /etc/X11/xorg.conf.d/99-calibration.conf
^G Get Help M-D DOS Format M-A Append M-B Backup File
^C Cancel M-M Mac Format M-P Prepend

You can now try to run X again with

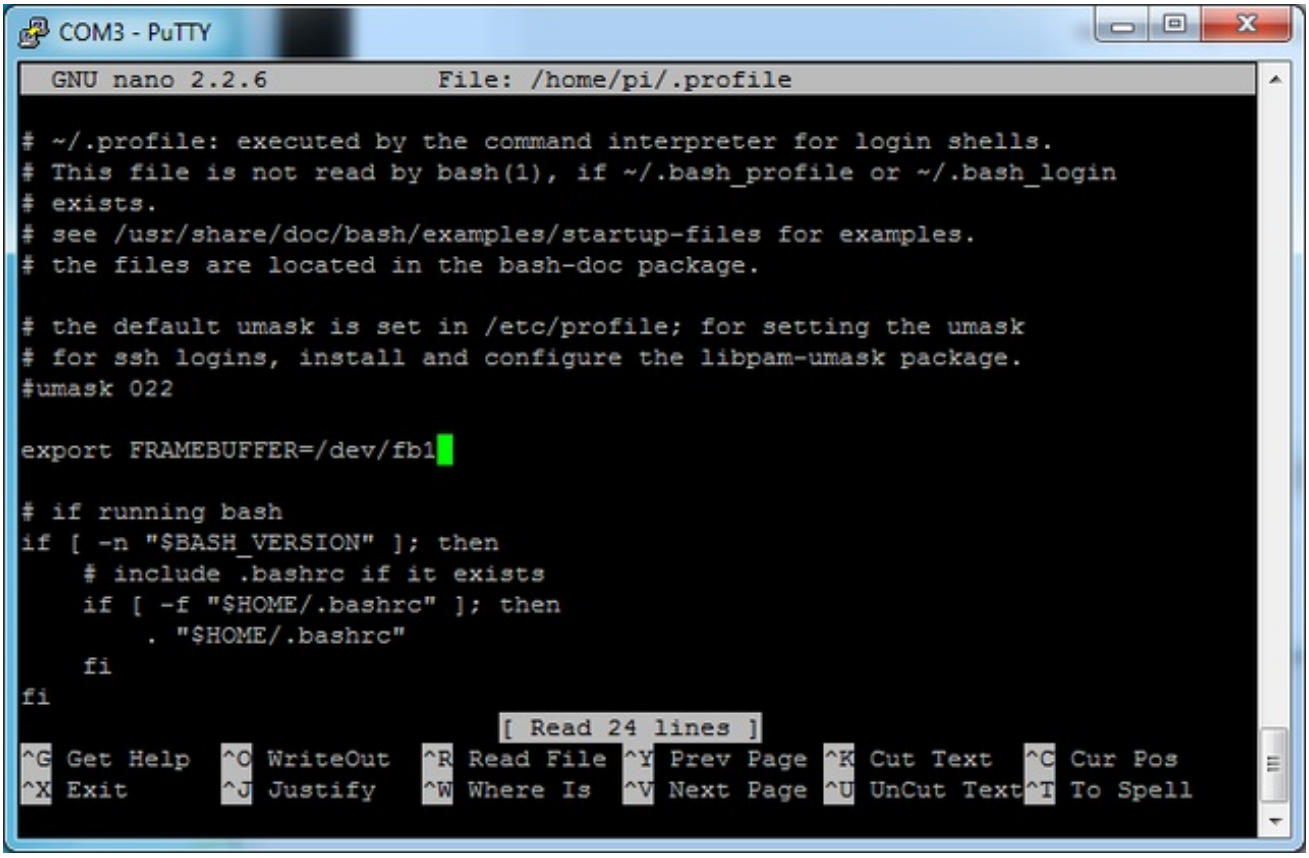
> **FRAMEBUFFER=/dev/fb1 startx**

Type Control-C to quit **X**

If you don't ever want to have to type FRAMEBUFFER=/dev/fb1 before startx, you can make it a default state by editing your profile file: **sudo nano ~/.profile** and adding

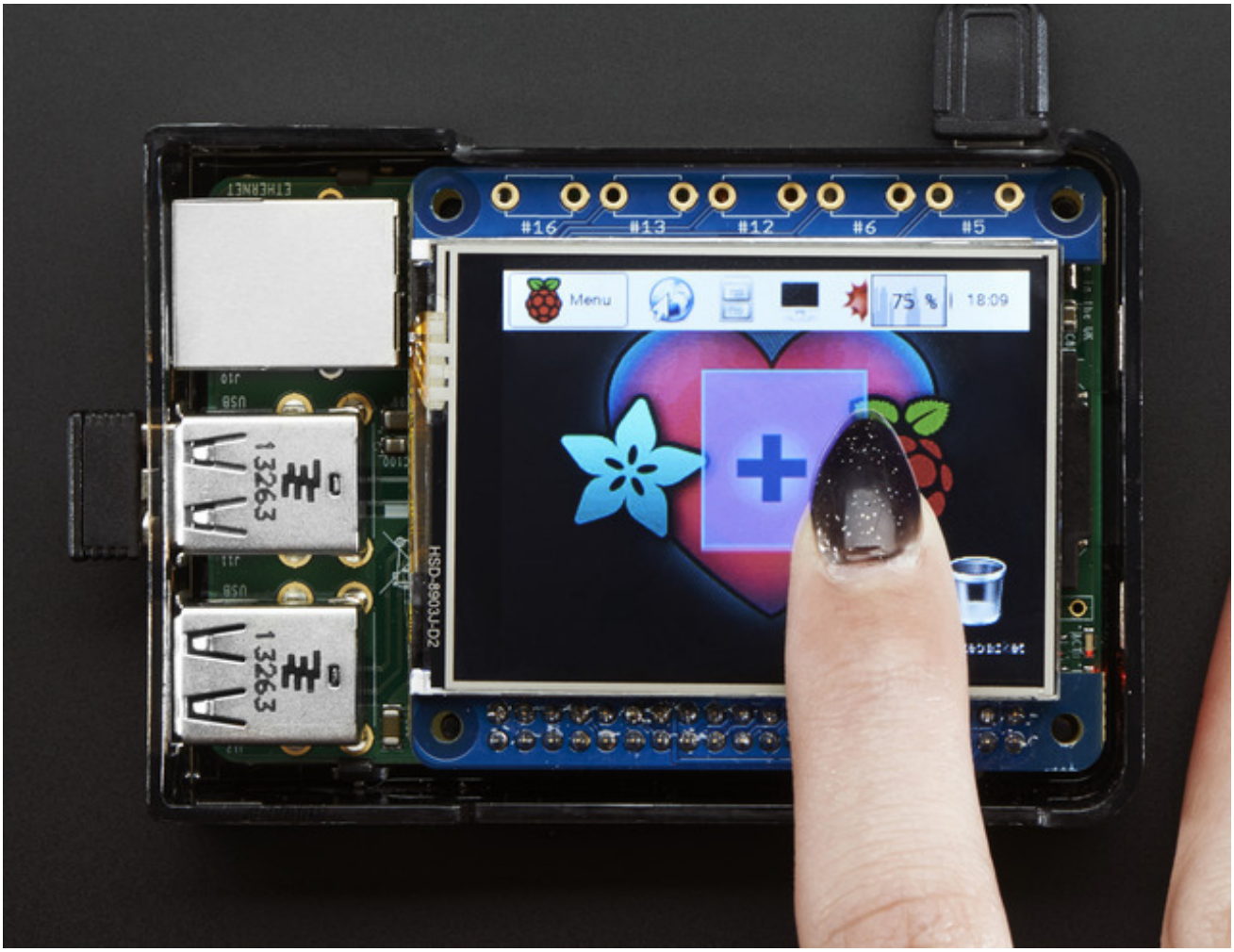
> **export FRAMEBUFFER=/dev/fb1**

near the top and saving the file. Then reboot to reload the profile file. It will now always assume you want to use /dev/fb1


COM3 - PuTTY
GNU nano 2.2.6 File: /home/pi/.profile
~/.profile: executed by the command interpreter for login shells.
This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
exists.
see /usr/share/doc/bash/examples/startup-files for examples.
the files are located in the bash-doc package.
the default umask is set in /etc/profile; for setting the umask
for ssh logins, install and configure the libpam-umask package.
#umask 022
export FRAMEBUFFER=/dev/fb1
# if running bash
if [-n "$BASH_VERSION"]; then
# include .bashrc if it exists
if [-f "$HOME/.bashrc"]; then
. "$HOME/.bashrc"
fi
fi
[Read 24 lines]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

Touchscreen Install & Calibrate

If you've grabbed our Easy Install image, or use the script, this step is not required, it's already done! This is just for advanced users who are curious on how to configure and customize the touchscreen



Setting up the Touchscreen

Now that the screen is working nicely, we'll take care of the touchscreen. There's just a bit of calibration to do, but it isn't hard at all.

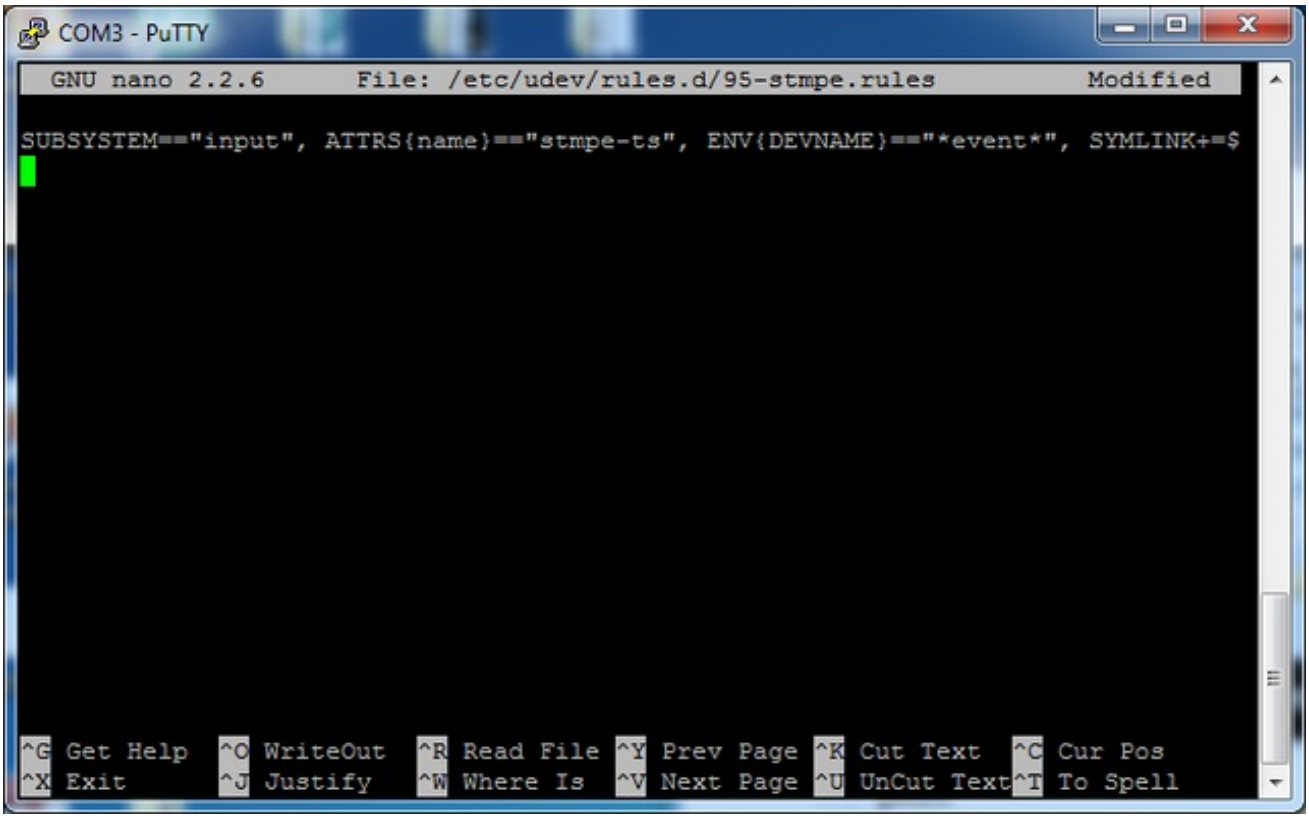
Before we start, we'll make a **udev** rule for the touchscreen. That's because the **eventX** name of the device will change a lot and its annoying to figure out what its called depending on whether you have a keyboard or other mouse installed.

Run

> **sudo nano /etc/udev/rules.d/95-stmpe.rules**

to create a new **udev** file and copy & paste the following line in:

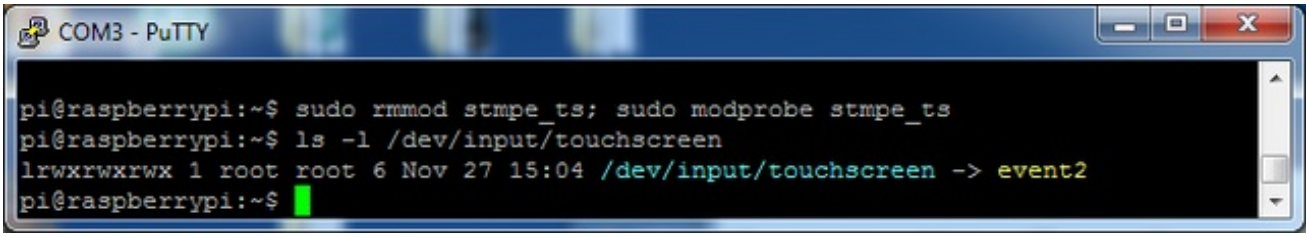
```
SUBSYSTEM=="input", ATTRS{name}=="stmpe-ts", ENV{DEVNAME}=="*event*", SYMLINK+="input/touchsc
```


COM3 - PuTTY
GNU nano 2.2.6 File: /etc/udev/rules.d/95-stmpe.rules Modified
SUBSYSTEM=="input", ATTRS{name}=="stmpe-ts", ENV{DEVNAME}=="*event*", SYMLINK+=$
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

Remove and re-install the touchscreen with

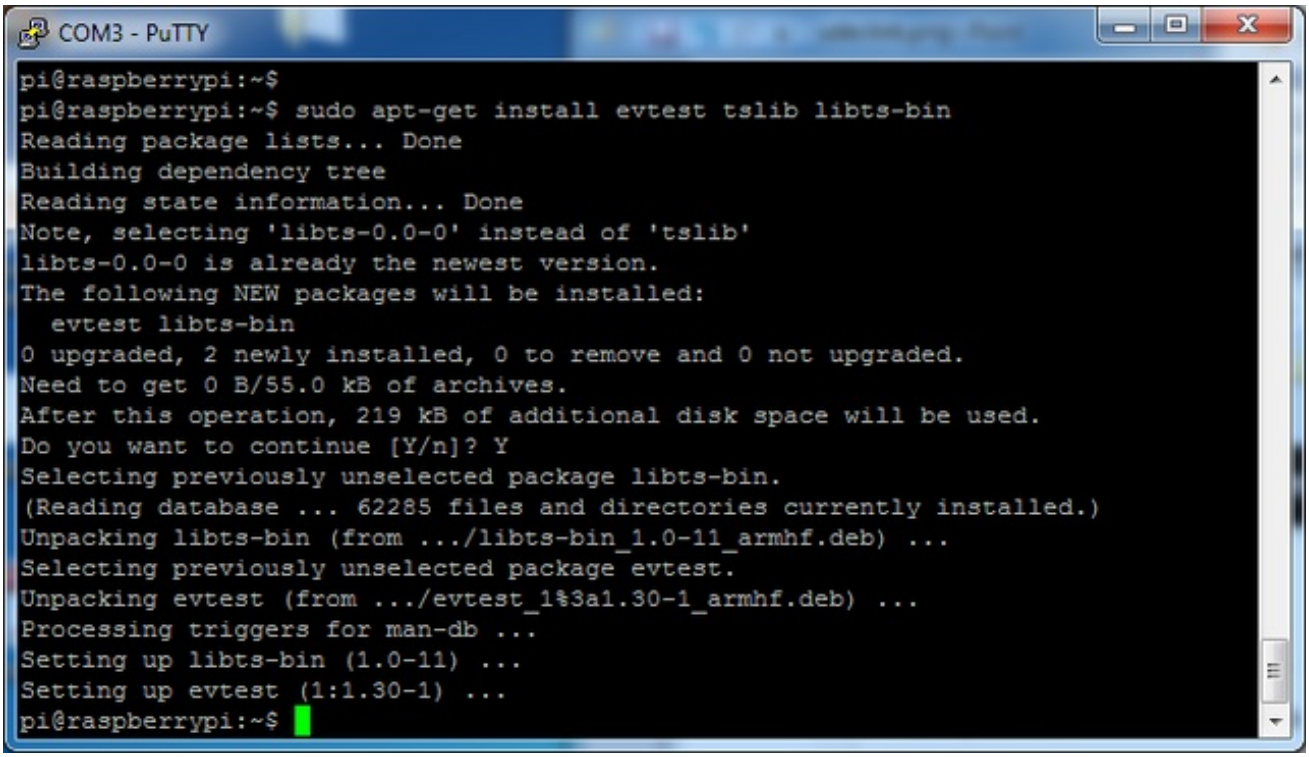
> **sudo rmmod stmpe_ts; sudo modprobe stmpe_ts**

Then type **ls -l /dev/input/touchscreen**
It should point to **eventX** where X is some number, that number will be different on different setups since other keyboards/mice/USB devices will take up an event slot


COM3 - PuTTY
pi@raspberrypi:~$ sudo rmmod stmpe_ts; sudo modprobe stmpe_ts
pi@raspberrypi:~$ ls -l /dev/input/touchscreen
lrwxrwxrwx 1 root root 6 Nov 27 15:04 /dev/input/touchscreen -> event2
pi@raspberrypi:~$

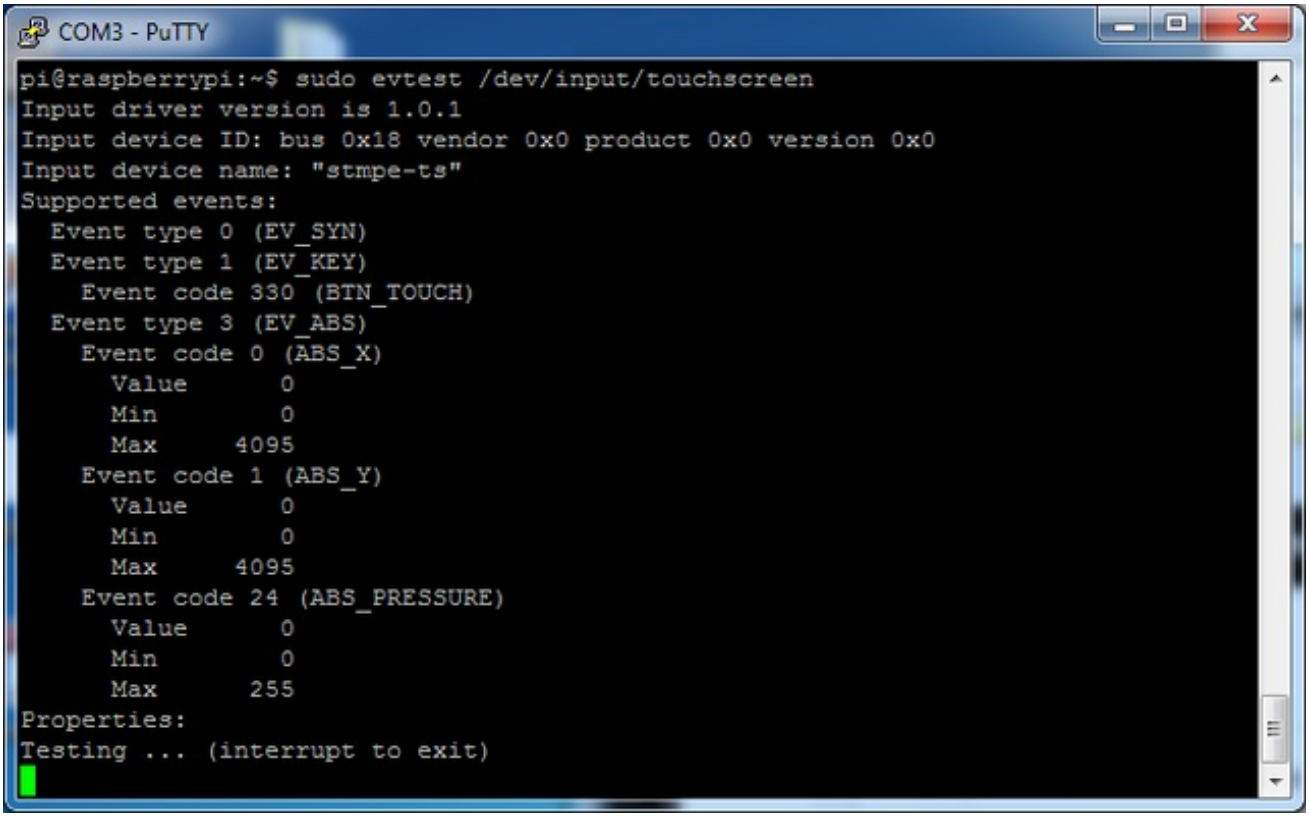
There are some tools we can use to calibrate & debug the touchscreen. Install the "event test" and "touchscreen library" packages with

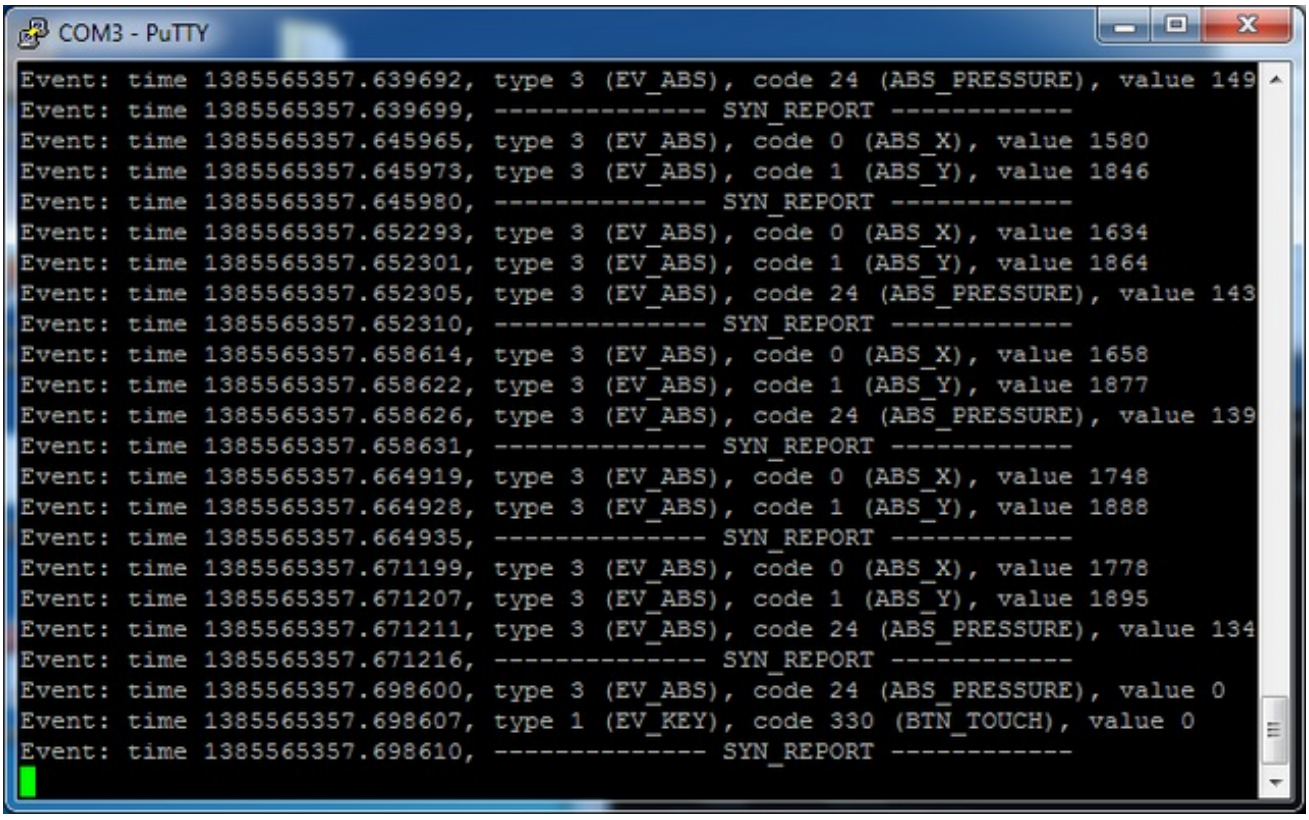
> **sudo apt-get install evtest tslib libts-bin**


COM3 - PuTTY
pi@raspberrypi:~$
pi@raspberrypi:~$ sudo apt-get install evtest tslib libts-bin
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libts-0.0-0' instead of 'tslib'
libts-0.0-0 is already the newest version.
The following NEW packages will be installed:
evtest libts-bin
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/55.0 kB of archives.
After this operation, 219 kB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Selecting previously unselected package libts-bin.
(Reading database ... 62285 files and directories currently installed.)
Unpacking libts-bin (from .../libts-bin_1.0-11_armhf.deb) ...
Selecting previously unselected package evtest.
Unpacking evtest (from .../evtest_1%3a1.30-1_armhf.deb) ...
Processing triggers for man-db ...
Setting up libts-bin (1.0-11) ...
Setting up evtest (1:1.30-1) ...
pi@raspberrypi:~$

Running evtest

Now you can use some tools such as **sudo evtest /dev/input/touchscreen** which will let you see touchscreen events in real time, press on the touchscreen to see the reports.



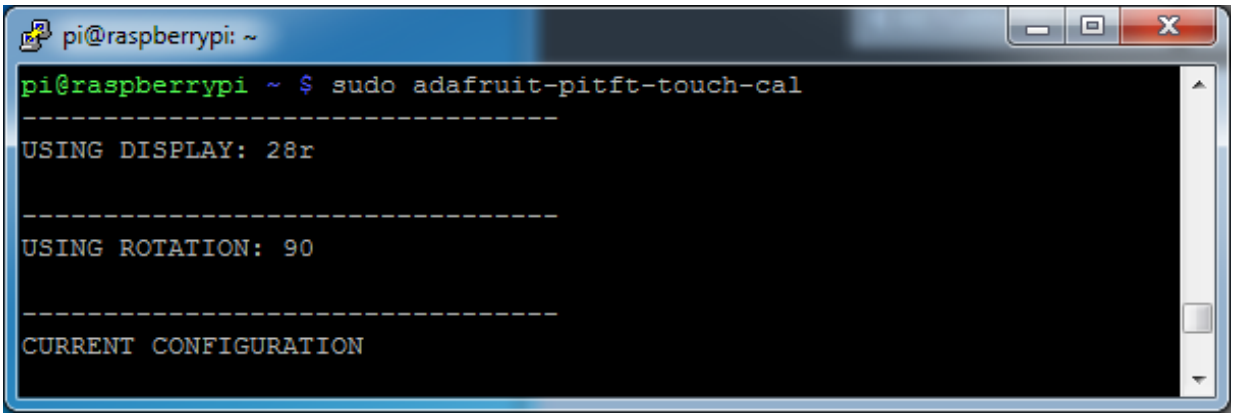

COM3 - PuTTY
Event: time 1385565357.639692, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 149
Event: time 1385565357.639699, ---------- SYN_REPORT ----------
Event: time 1385565357.645965, type 3 (EV_ABS), code 0 (ABS_X), value 1580
Event: time 1385565357.645973, type 3 (EV_ABS), code 1 (ABS_Y), value 1846
Event: time 1385565357.645980, ---------- SYN_REPORT ----------
Event: time 1385565357.652293, type 3 (EV_ABS), code 0 (ABS_X), value 1634
Event: time 1385565357.652301, type 3 (EV_ABS), code 1 (ABS_Y), value 1864
Event: time 1385565357.652305, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 143
Event: time 1385565357.652310, ---------- SYN_REPORT ----------
Event: time 1385565357.658614, type 3 (EV_ABS), code 0 (ABS_X), value 1658
Event: time 1385565357.658622, type 3 (EV_ABS), code 1 (ABS_Y), value 1877
Event: time 1385565357.658626, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 139
Event: time 1385565357.658631, ---------- SYN_REPORT ----------
Event: time 1385565357.664919, type 3 (EV_ABS), code 0 (ABS_X), value 1748
Event: time 1385565357.664928, type 3 (EV_ABS), code 1 (ABS_Y), value 1888
Event: time 1385565357.664935, ---------- SYN_REPORT ----------
Event: time 1385565357.671199, type 3 (EV_ABS), code 0 (ABS_X), value 1778
Event: time 1385565357.671207, type 3 (EV_ABS), code 1 (ABS_Y), value 1895
Event: time 1385565357.671211, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 134
Event: time 1385565357.671216, ---------- SYN_REPORT ----------
Event: time 1385565357.698600, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 0
Event: time 1385565357.698607, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0
Event: time 1385565357.698610, ---------- SYN_REPORT ----------

AutoMagic Calibration Script

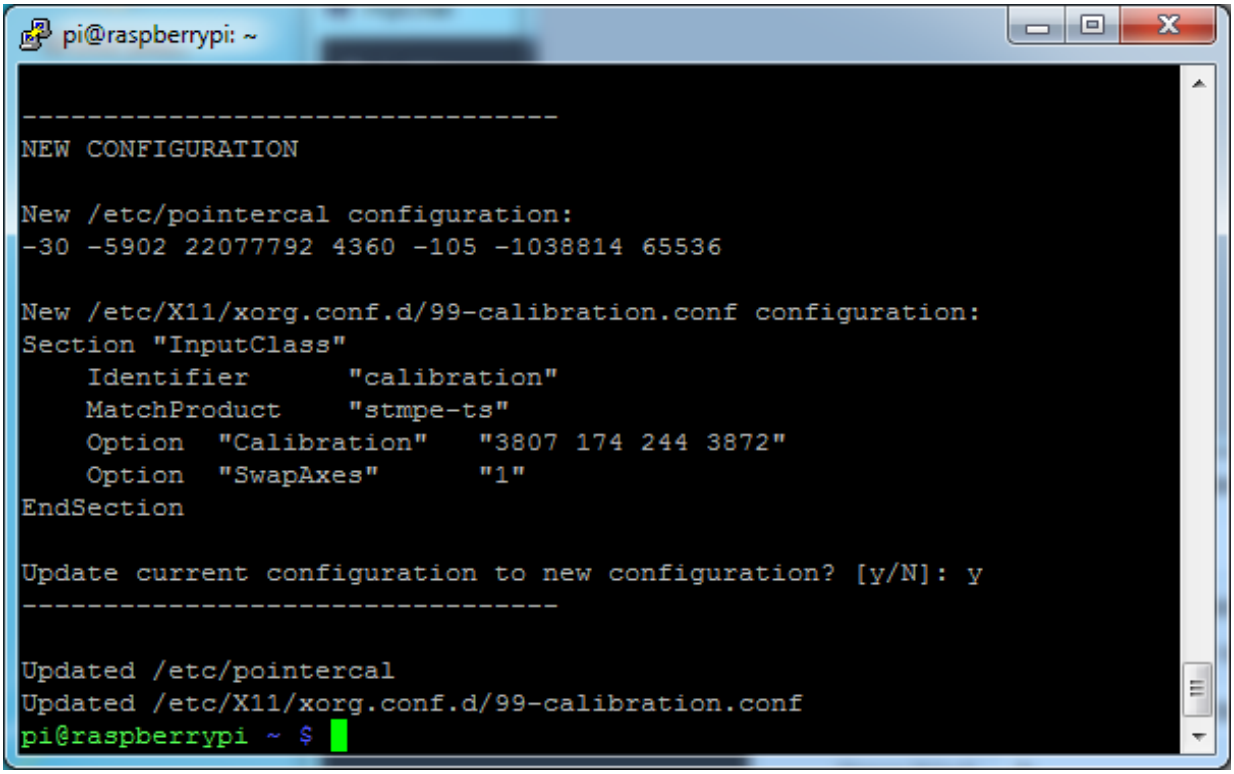
If you rotate the display you need to recalibrate the touchscreen to work with the new screen orientation. You can manually run the calibration processes in the next section, or you can run a small Python script which will automatically set a default touchscreen calibration based on the screen orientation.

This helper is automatically installed for you but if you'd like you can check it out here on github (http://adafru.it/elu)

Run it at the command line with **sudo adafruit-pitft-touch-cal**
it will try to figure out what display you have installed and the rotation it's set up for



By default the script will attempt to read the screen orientation by examining the PiTFT module configuration with modprobe. If the script can read the orientation it will print out the current orientation, the current touchscreen calibration values, and the new touchscreen calibration values baesd on the current orientation. Before updating the calibration the script will ask you to confirm that you'd like to make the change. Press **y** and enter to confirm.



Try using this default calibration script to easily calibrate your touchscreen display. Note that the calibration values might not be exactly right for your display, but they should be close enough for most needs. If you need the most accurate touchscreen calibration, follow the steps in the next section to manually calibrate the touchscreen.

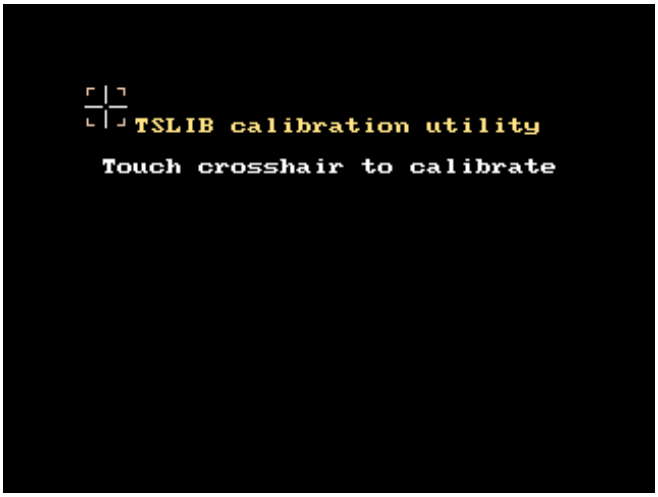
Manual Calibration

If the "automagic" calibration technique isn't working for you, or you have some other setup where you need to carefully calibrate you can do it 'manually'

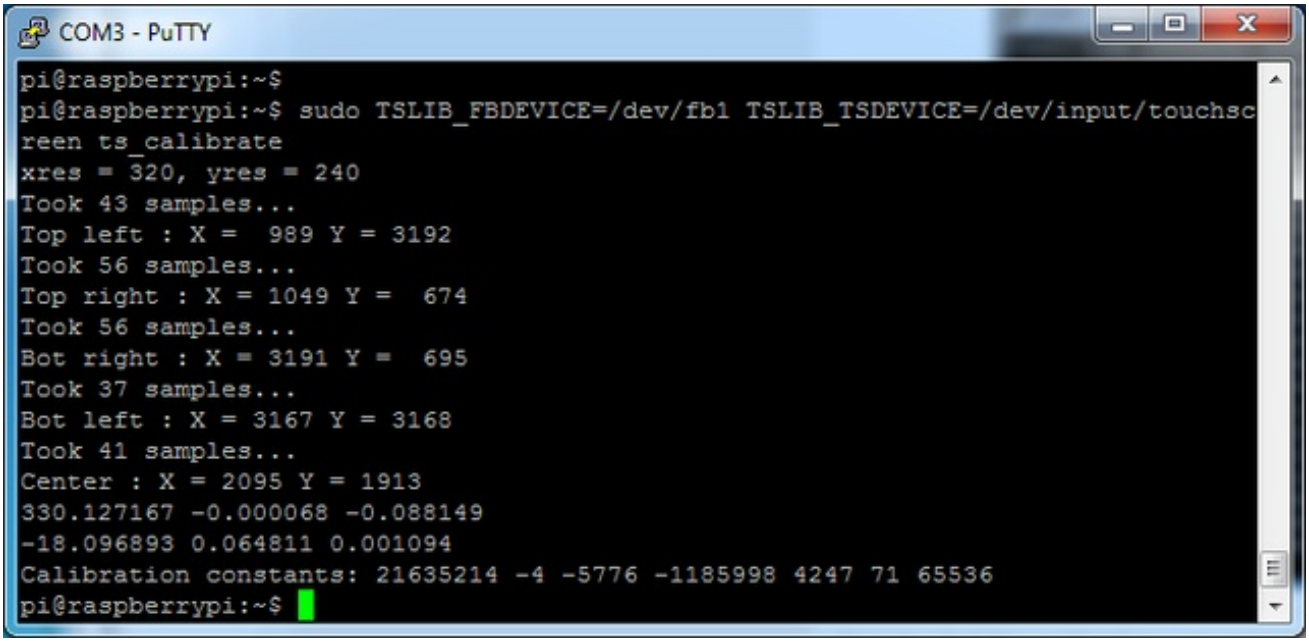
You will want to calibrate the screen once but shouldn't have to do it more than that. We'll begin by calibrating on the command line by running

```
sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_TSDEVICE=/dev/input/touchscreen
ts_calibrate
```

follow the directions on the screen, touching each point. Using a stylus is suggested so you get a precise touch. Don't use something metal, plastic only!



You should see five crosshair targets. If you see less than that, the touchscreen probably generated multiple signals for a single touch, and you should try calibrating again.


COM3 - PuTTY
pi@raspberrypi:~$
pi@raspberrypi:~$ sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_TSDEVICE=/dev/input/touchsc
reen ts_calibrate
xres = 320, yres = 240
Took 43 samples...
Top left : X = 989 Y = 3192
Took 56 samples...
Top right : X = 1049 Y = 674
Took 56 samples...
Bot right : X = 3191 Y = 695
Took 37 samples...
Bot left : X = 3167 Y = 3168
Took 41 samples...
Center : X = 2095 Y = 1913
330.127167 -0.000068 -0.088149
-18.096893 0.064811 0.001094
Calibration constants: 21635214 -4 -5776 -1185998 4247 71 65536
pi@raspberrypi:~$

Next you can run **sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_TSDEVICE=/dev/input/touchscreen ts_test** which will let you draw-test the touch screen. Go back and re-calibrate if you feel the screen isn't precise enough!

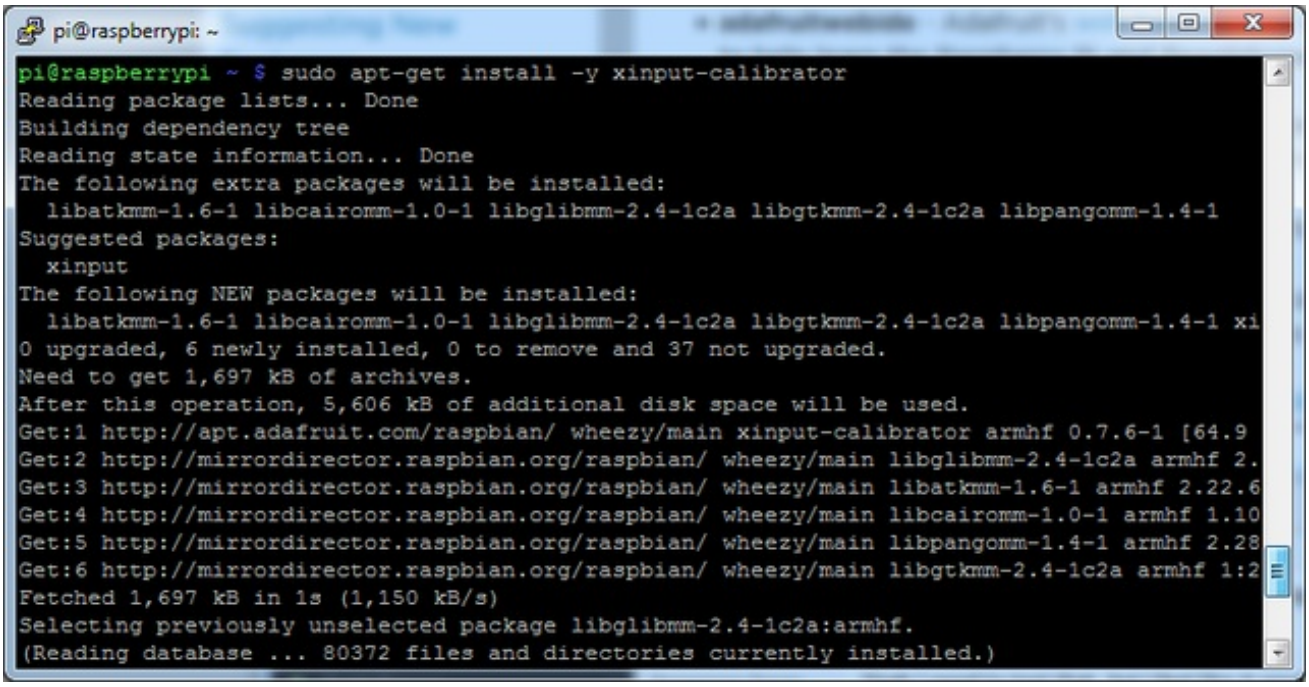
X Calibration

You can also calibrate the X input system but you have to use a different program called **xinput_calibrator**

You can do this if the calibration on the screen isnt to your liking or any time you change the **rotate=XX** module settings for the screen. Since the screen and touch driver are completely separated, the touchscreen doesn't auto-rotate

Normally you'd have to compile it but we have a ready to go package for you so run:

```
sudo apt-get install -y xinput-calibrator
```


pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo apt-get install -y xinput-calibrator
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
libatkmm-1.6-1 libcairomm-1.0-1 libglibmm-2.4-1c2a libgtkmm-2.4-1c2a libpangomm-1.4-1
Suggested packages:
xinput
The following NEW packages will be installed:
libatkmm-1.6-1 libcairomm-1.0-1 libglibmm-2.4-1c2a libgtkmm-2.4-1c2a libpangomm-1.4-1 xi
0 upgraded, 6 newly installed, 0 to remove and 37 not upgraded.
Need to get 1,697 kB of archives.
After this operation, 5,606 kB of additional disk space will be used.
Get:1 http://apt.adafruit.com/raspbian/ wheezy/main xinput-calibrator armhf 0.7.6-1 [64.9
Get:2 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libglibmm-2.4-1c2a armhf 2.
Get:3 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libatkmm-1.6-1 armhf 2.22.6
Get:4 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libcairomm-1.0-1 armhf 1.10
Get:5 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libpangomm-1.4-1 armhf 2.28
Get:6 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libgtkmm-2.4-1c2a armhf 1:2
Fetched 1,697 kB in 1s (1,150 kB/s)
Selecting previously unselected package libglibmm-2.4-1c2a:armhf.
(Reading database ... 80372 files and directories currently installed.)

Before you start the xinput_calibrator you will need to delete the old calibration data so run

```
sudo rm /etc/X11/xorg.conf.d/99-calibration.conf
```

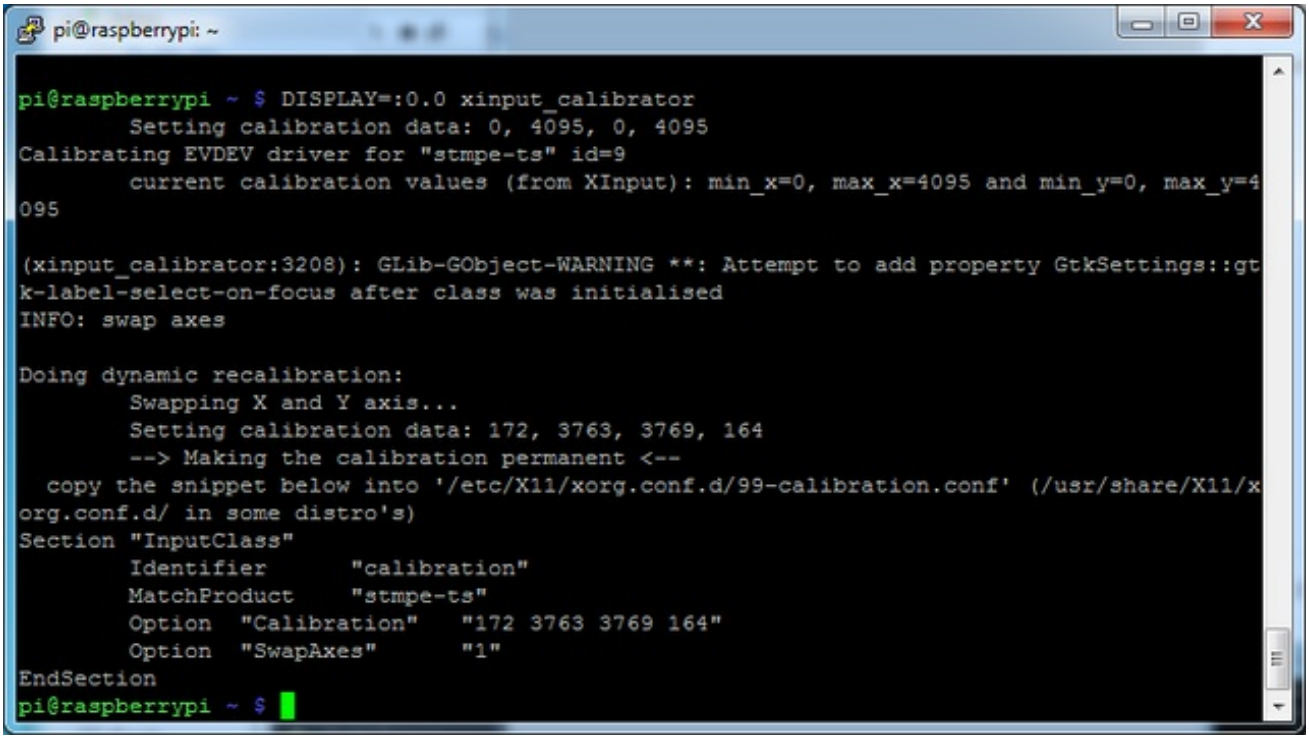
Before running **startx** and the calibrator - otherwise it gets really confused!

Now you'll have to run the xcalibrator while also running X. You can do this by **startx** and then opening up the terminal program and running the **xinput_calibrator** command (which is challenging to do on such a small screen) OR you can do what we do which is run startx in a SSH/Terminal shell and then run the xinput_calibrator from the same shell, which requires the following command order:

```
FRAMEBUFFER=/dev/fb1 startx &
DISPLAY=:0.0 xinput_calibrator
```

Follow the directions on screen

Once complete you'll get something like:


pi@raspberrypi: ~
pi@raspberrypi ~ $ DISPLAY=:0.0 xinput_calibrator
Setting calibration data: 0, 4095, 0, 4095
Calibrating EVDEV driver for "stmpe-ts" id=9
current calibration values (from XInput): min_x=0, max_x=4095 and min_y=0, max_y=4095
(xinput_calibrator:3208): GLib-GObject-WARNING **: Attempt to add property GtkSettings::gtk-label-select-on-focus after class was initialised
INFO: swap axes
Doing dynamic recalibration:
Swapping X and Y axis...
Setting calibration data: 172, 3763, 3769, 164
--> Making the calibration permanent <--
copy the snippet below into '/etc/X11/xorg.conf.d/99-calibration.conf' (/usr/share/X11/xorg.conf.d/ in some distro's)
Section "InputClass"
Identifier "calibration"
MatchProduct "stmpe-ts"
Option "Calibration" "172 3763 3769 164"
Option "SwapAxes" "1"
EndSection
pi@raspberrypi ~ $

Run **sudo nano /etc/X11/xorg.conf.d/99-calibration.conf** and copy the

```
Section "InputClass"
        Identifier      "calibration"
        MatchProduct    "stmpe-ts"
        Option  "Calibration"   "172 3763 3769 164"
        Option  "SwapAxes"      "1"
EndSection
```

or whatever you got, into there. You can quit X if you want by typing **fg** to bring that command into the foreground, and then Control-C to quit.

Depending on the 'rotation' of the screen, when you do this calibration, you may need to comment out the SwapAxes part with a # and/or **swap the numbers around so looks like:**

Option "Calibration" "119 3736 3850 174"

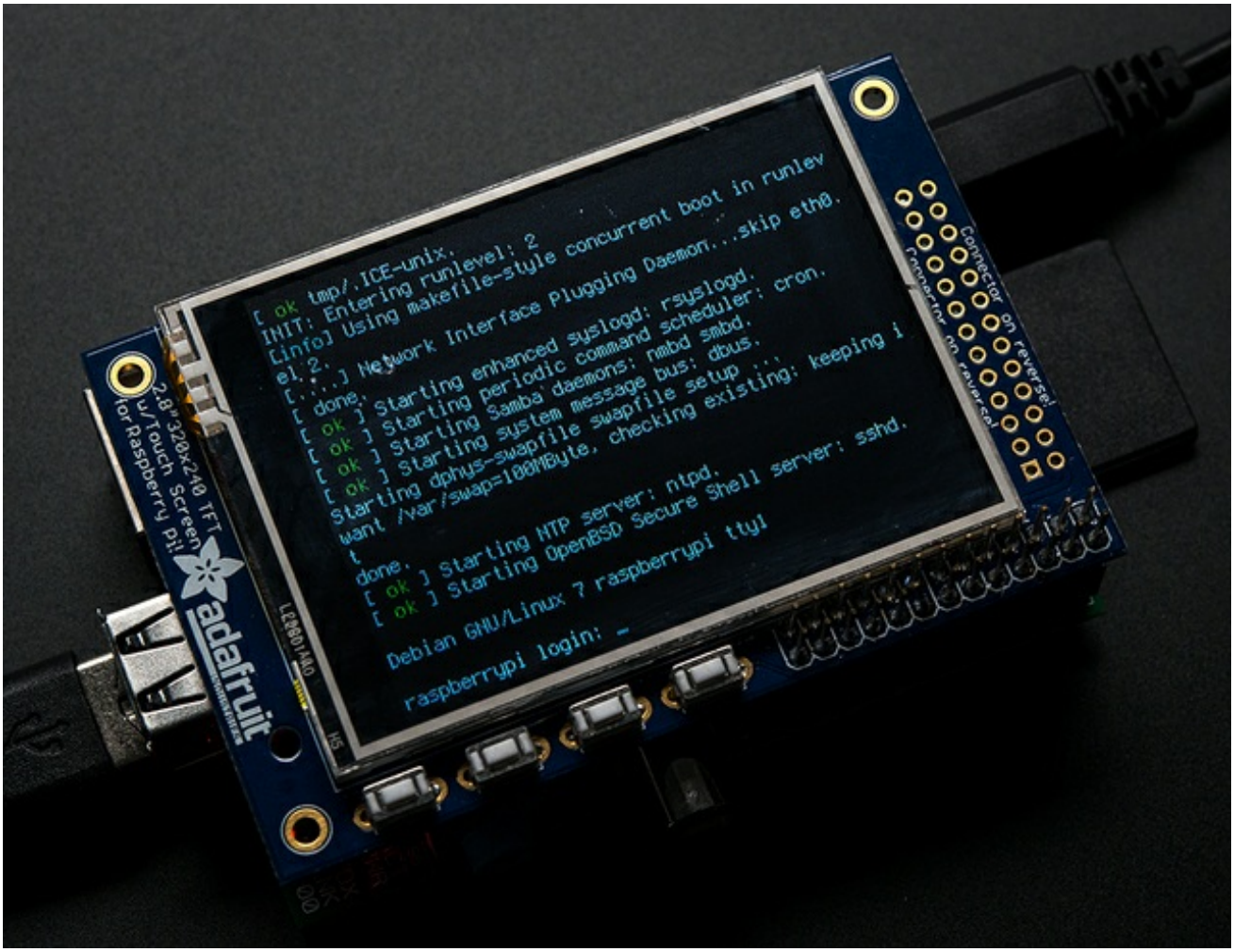
to

Option "Calibration" "3736 119 174 3850"

Your touchscreen is now super calibrated, hurrah!

Using the Console

If you've grabbed our Easy Install image, or use the script, this step is not required, it's already done! This is just for advanced users who are curious on how to configure and customize the console


[ok] tmp/.ICE-unix.
INIT: Entering runlevel: 2
[info] Using makefile-style concurrent boot in runlev
el 2.
[....] Network Interface Plugging Daemon...skip eth0.
[done.
[ok] Starting enhanced syslogd: rsyslogd.
[ok] Starting periodic command scheduler: cron.
[ok] Starting Samba daemons: nmbd smbd.
[ok] Starting system message bus: dbus.
Starting dphys-swapfile swapfile setup ...
want /var/swap=100MByte, checking existing: keeping i
t
done.
[ok] Starting HTP server: ntpd.
[ok] Starting OpenBSD Secure Shell server: sshd.
Debian GNU/Linux 7 raspberrypi tty1
raspberrypi login: _
adafruit
2.8" 320x240 TFT w/Touch Screen for Raspberry Pi!
Connector on reverse!

One fun thing you can do with the display is have it as your main console instead of the HDMI/TV output. Even though it is small, with a good font you can get 20 x 40 of text. For more details, check out https://github.com/notro/fbtft/wiki/Boot-console (http://adafru.it/cXQ)

First up, we'll update the boot configuration file to use the TFT framebuffer /**dev/fb1** instead of the HDMI/TV framebuffer /dev/fb0

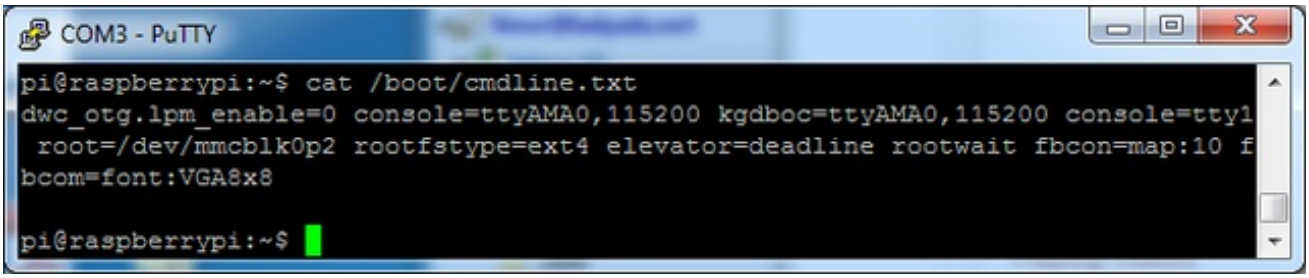
> **sudo nano /boot/cmdline.txt**

you can also edit it by putting the SD card into a computer and opening the same file.

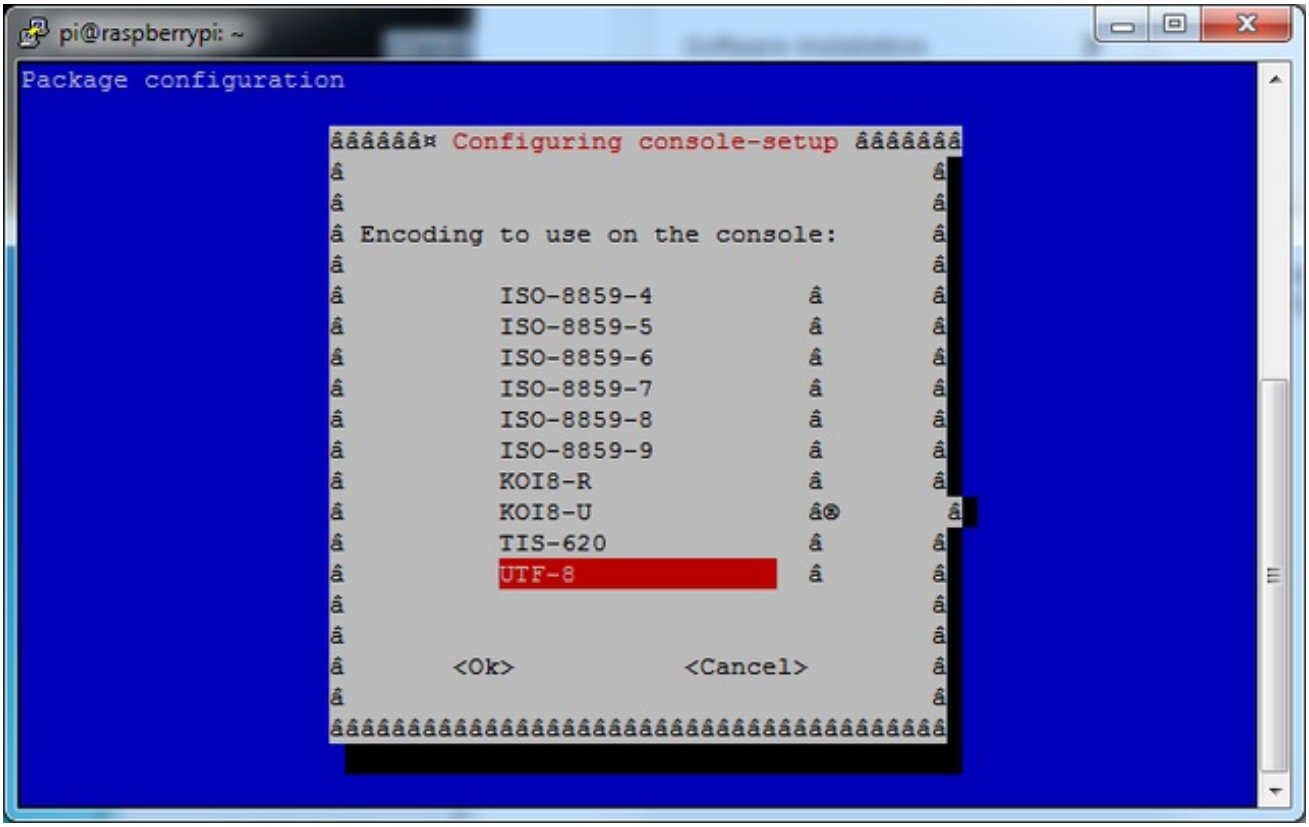
At the end of the line, find the text that says **rootwait** and right after that, enter in: **fbcon=map:10 fbcon=font:VGA8x8** then save the file.

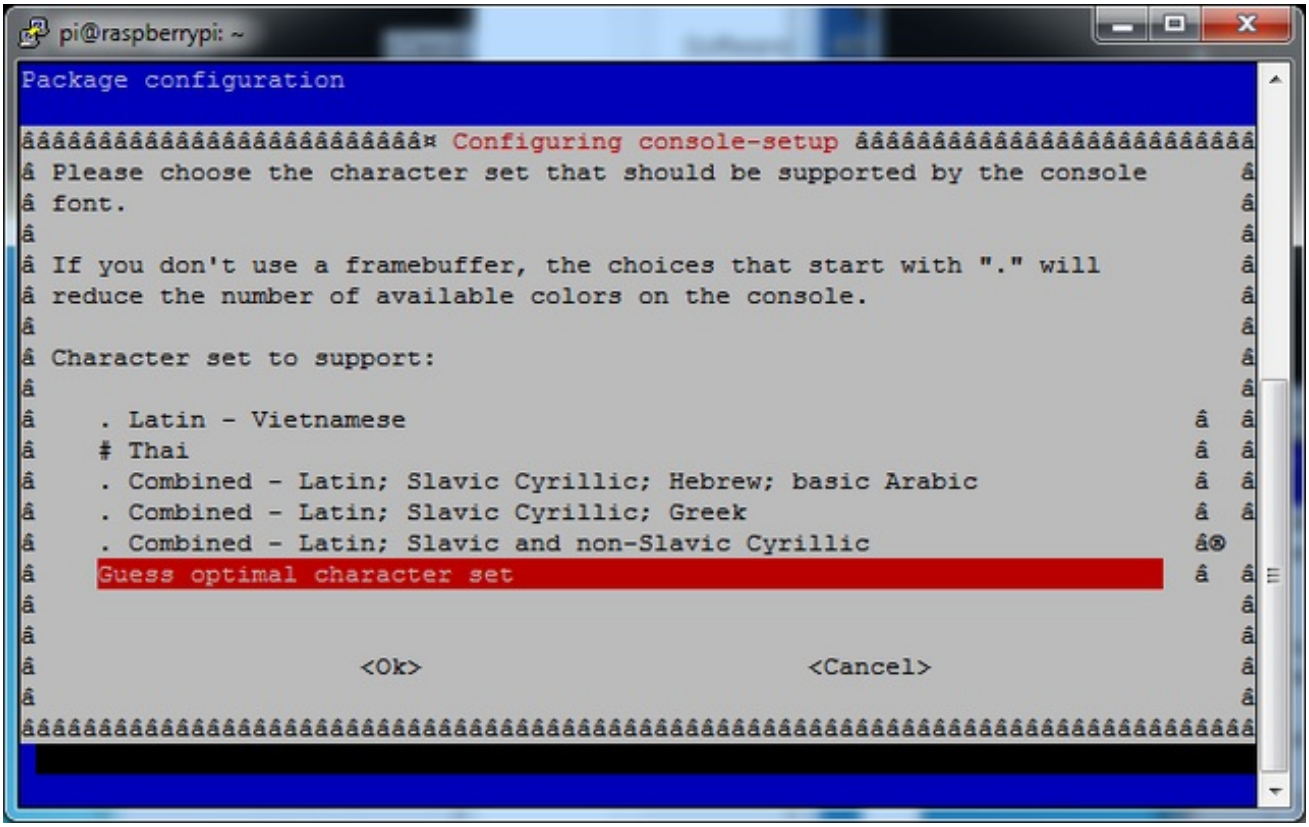
On the next boot, it will bring up the console.

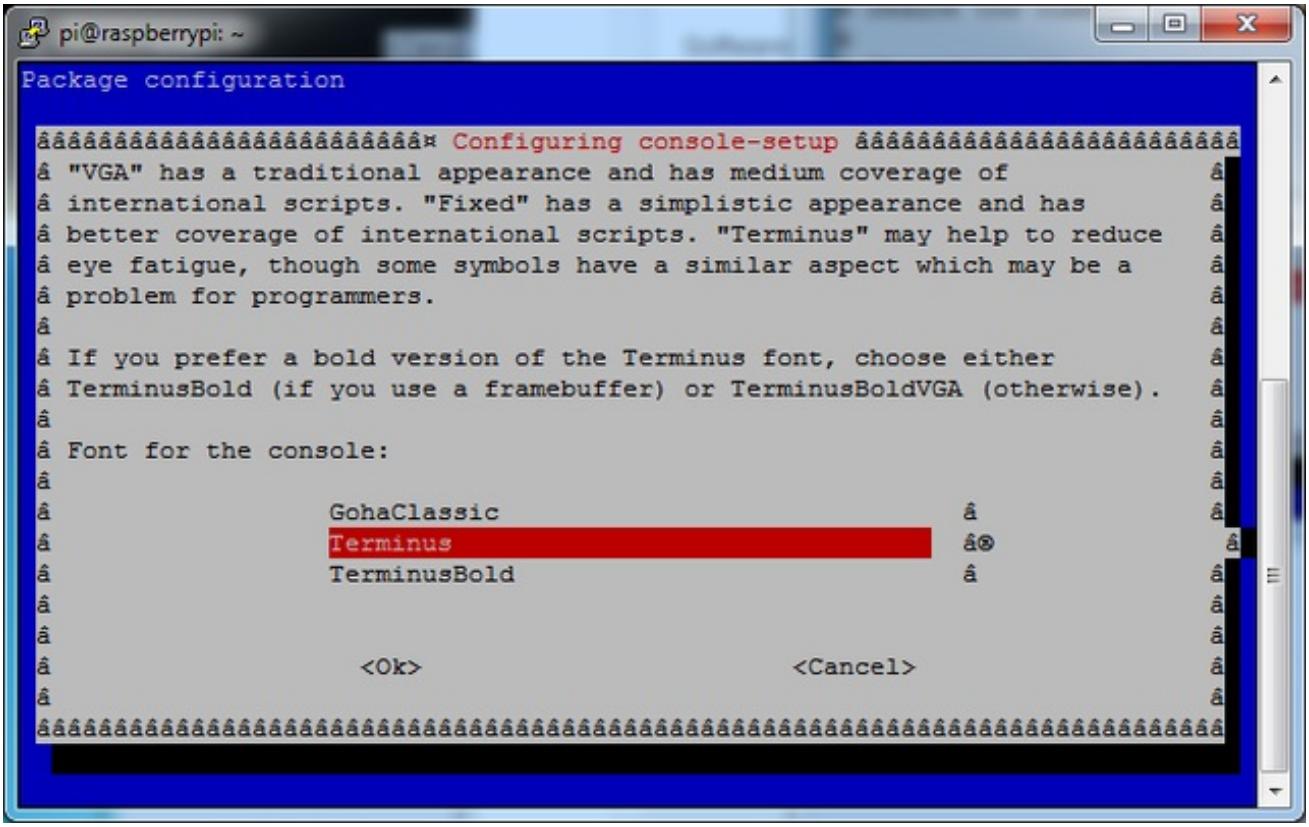
Note that the kernel has to load up the display driver module before it can display anything on it so you won't get the rainbow screen, a NooBs prompt, or a big chunk of the kernel details since the module is loaded fairly late in the boot process.

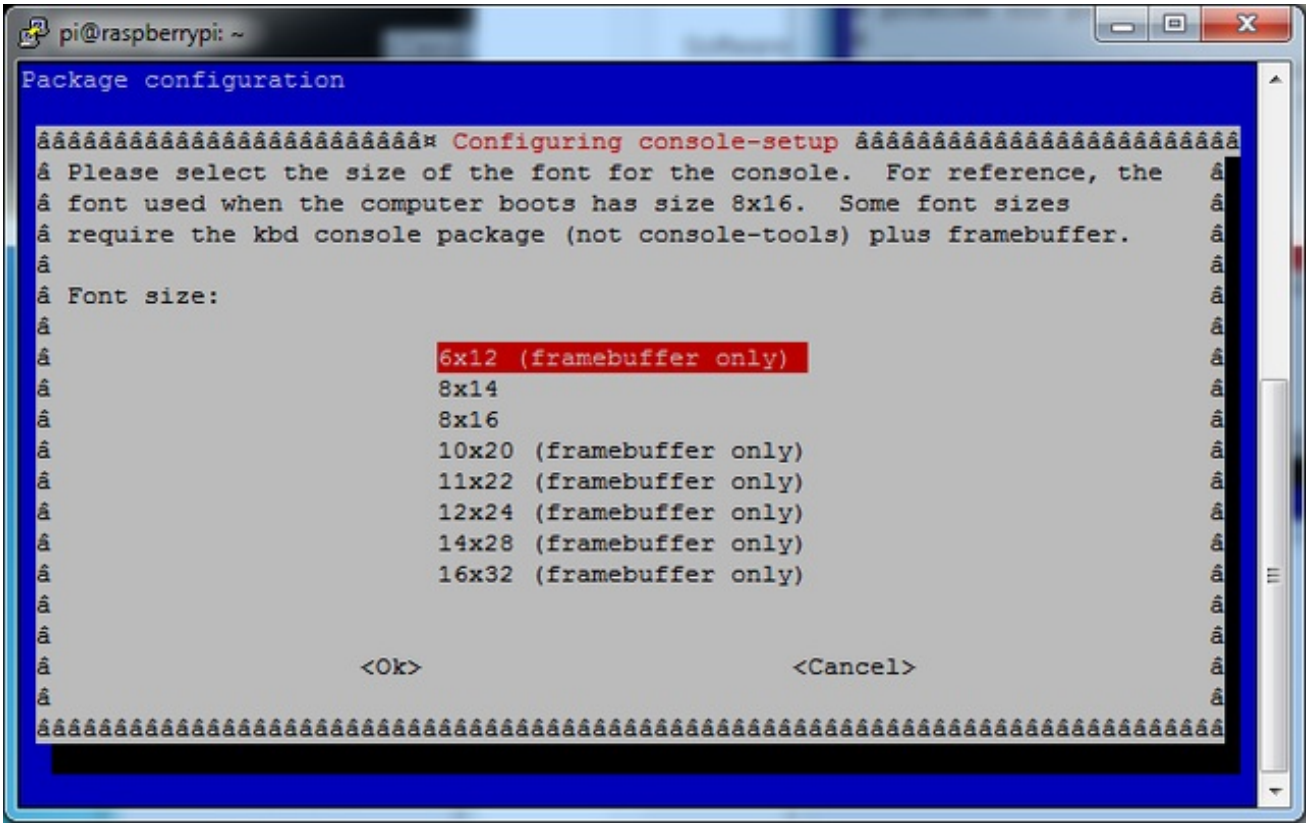


I think the VGA8x8 font is a bit chunky, you probably want 12x6 which is what is shown in the photo above. To change the font, run **sudo dpkg-reconfigure console-setup** and go thru to select Terminus 6x12




pi@raspberrypi: ~
Package configuration
Configuring console-setup
Please choose the character set that should be supported by the console font.
If you don't use a framebuffer, the choices that start with "." will reduce the number of available colors on the console.
Character set to support:
. Latin - Vietnamese
Thai
. Combined - Latin; Slavic Cyrillic; Hebrew; basic Arabic
. Combined - Latin; Slavic Cyrillic; Greek
. Combined - Latin; Slavic and non-Slavic Cyrillic
Guess optimal character set
<Ok>
<Cancel>


pi@raspberrypi: ~
Package configuration
Configuring console-setup
"VGA" has a traditional appearance and has medium coverage of international scripts. "Fixed" has a simplistic appearance and has better coverage of international scripts. "Terminus" may help to reduce eye fatigue, though some symbols have a similar aspect which may be a problem for programmers.
If you prefer a bold version of the Terminus font, choose either TerminusBold (if you use a framebuffer) or TerminusBoldVGA (otherwise).
Font for the console:
GohaClassic
Terminus
TerminusBold
<Ok>
<Cancel>


pi@raspberrypi: ~
Package configuration
Configuring console-setup
Please select the size of the font for the console. For reference, the font used when the computer boots has size 8x16. Some font sizes require the kbd console package (not console-tools) plus framebuffer.
Font size:
6x12 (framebuffer only)
8x14
8x16
10x20 (framebuffer only)
11x22 (framebuffer only)
12x24 (framebuffer only)
14x28 (framebuffer only)
16x32 (framebuffer only)
<Ok>
<Cancel>

Turn off Console Blanking

You may notice the console goes black after 30 minutes, this is a sort of 'power saving' or 'screensaver' feature. You can disable this by editing /**etc**/**kbd**/**config** and setting the blank time to 0 (which turns it off)

```
BLANK_TIME=30
```

#16
#13
#12
#6
#5
Starting dphys-swapfile swapfile setup .
..
want /var/swap=100MByte, checking existi
ng: keeping it
done.
[ok] Starting NTP server: ntpd.
[....] Starting OpenBSD Secure Shell ser
[ok sshd.
Raspbian GNU/Linux 7 raspberrypi tty1
raspberrypi login: pi
Password:
Last login: Mon Apr 27 17:51:08 UTC 2015
on tty1
Linux raspberrypi 3.18.8+ #2 PREEMPT Mon
Mar 9 14:05:46 UTC 2015 armv6l
The programs included with the Debian GN
U/Linux system are free software;
the exact distribution terms for each pr
ogram are described in the
individual files in /usr/share/doc/*/cop
yright.
Debian GNU/Linux comes with ABSOLUTELY N
O WARRANTY, to the extent
permitted by applicable law.
pi@raspberrypi ~ $ FRAMEBUFFER=/dev/fb1
startx -- -dpi 60_
HSD-8903L02

Playing Videos

How To Play Videos

You can play many types of videos on the screen, using mplayer you don't even need to run X and you can script the movies to play using Python. We'll show you how to just play one video for now. To demo, we'll use an mp4 of Big Buck Bunny for 320 pixel wide screens. Below we show you how to create/resize videos, but to make it easy, just download our version with:

> **wget http://adafruit-download.s3.amazonaws.com/bigbuckbunny320p.mp4** (http://adafru.it/cXR)

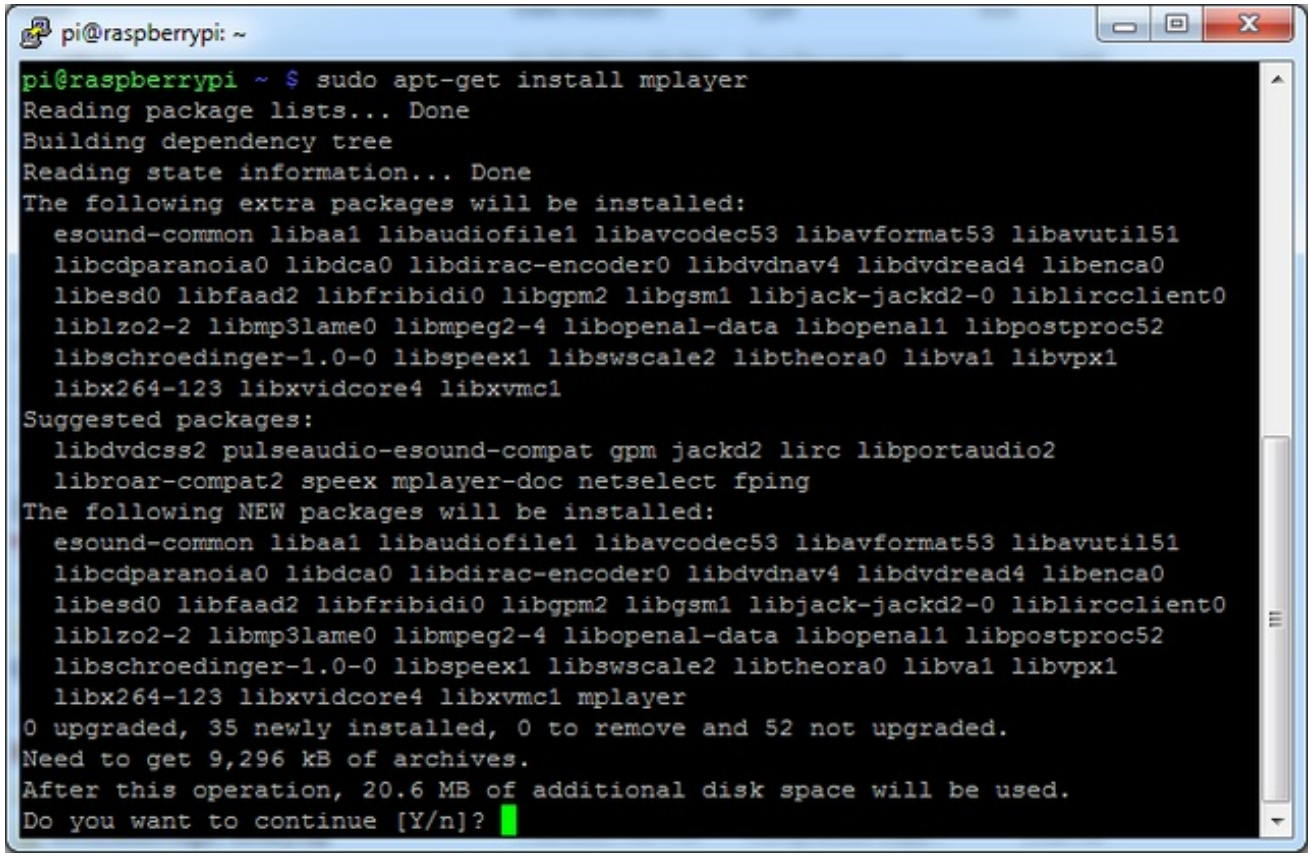
The video is 30MB which is a lot if you haven't expanded your SD card yet. Before you do this, run sudo raspi-config to expand the SD card so you don't run out of space!

If you don't have **mplayer** yet, run

sudo apt-get update

sudo apt-get install mplayer

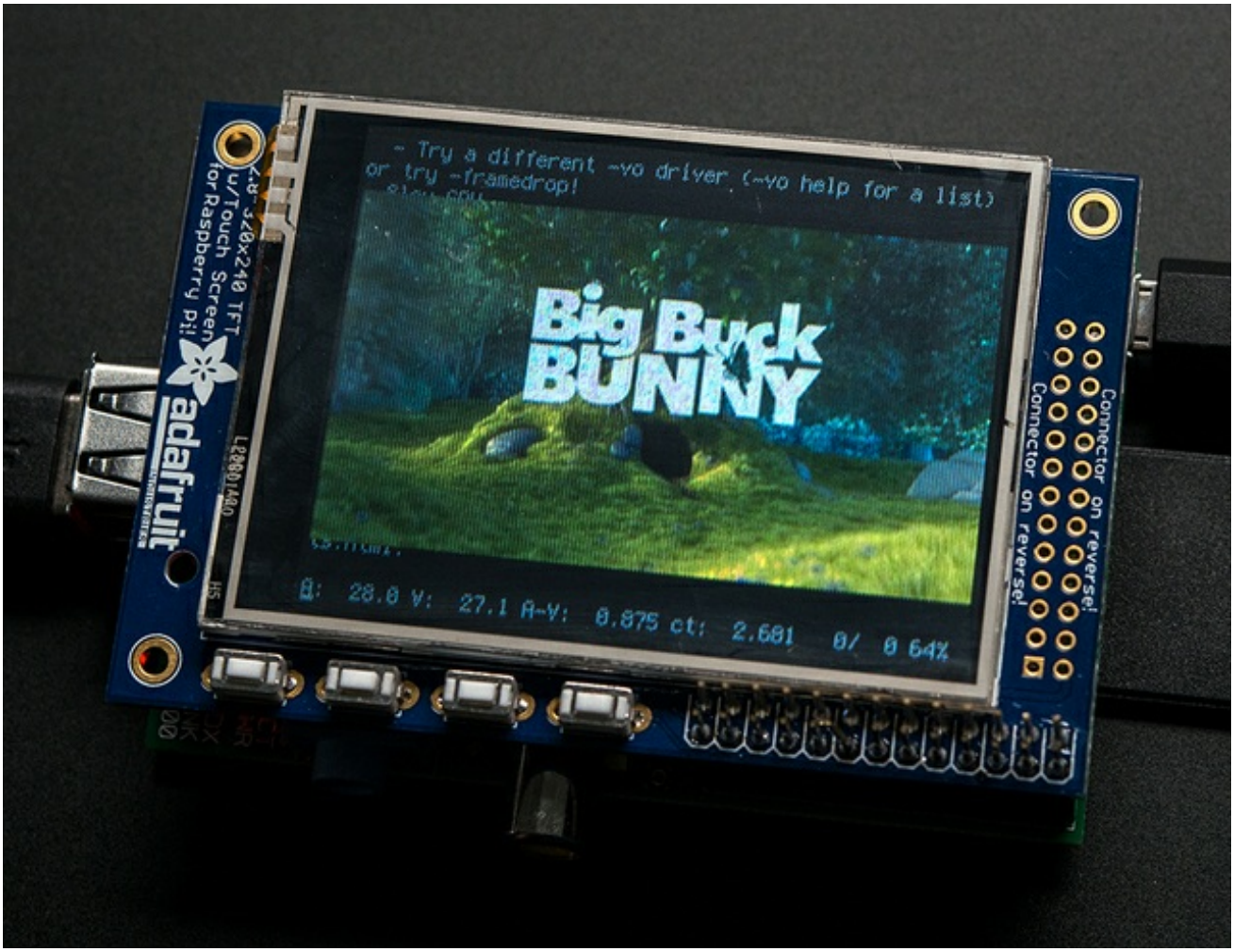
to install it. It may take a few minutes to complete


pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo apt-get install mplayer
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
esound-common libaa1 libaudiofile1 libavcodec53 libavformat53 libavutil51
libcdparanoia0 libdca0 libdirac-encoder0 libdvdnav4 libdvdread4 libenca0
libesd0 libfaad2 libfribidi0 libgpm2 libgsm1 libjack-jackd2-0 liblircclient0
liblzo2-2 libmp3lame0 libmpeg2-4 libopenal-data libopenal1 libpostproc52
libschroedinger-1.0-0 libspeex1 libswscale2 libtheora0 libva1 libvpx1
libx264-123 libxvidcore4 libxvmc1
Suggested packages:
libdvdcss2 pulseaudio-esound-compat gpm jackd2 lirc libportaudio2
libroar-compat2 speex mplayer-doc netselect fping
The following NEW packages will be installed:
esound-common libaa1 libaudiofile1 libavcodec53 libavformat53 libavutil51
libcdparanoia0 libdca0 libdirac-encoder0 libdvdnav4 libdvdread4 libenca0
libesd0 libfaad2 libfribidi0 libgpm2 libgsm1 libjack-jackd2-0 liblircclient0
liblzo2-2 libmp3lame0 libmpeg2-4 libopenal-data libopenal1 libpostproc52
libschroedinger-1.0-0 libspeex1 libswscale2 libtheora0 libva1 libvpx1
libx264-123 libxvidcore4 libxvmc1 mplayer
0 upgraded, 35 newly installed, 0 to remove and 52 not upgraded.
Need to get 9,296 kB of archives.
After this operation, 20.6 MB of additional disk space will be used.
Do you want to continue [Y/n]?

OK now you just have to run

mplayer -vo fbdev2:/dev/fb1 -x 240 -y 320 -framedrop bigbuckbunny320p.mp4

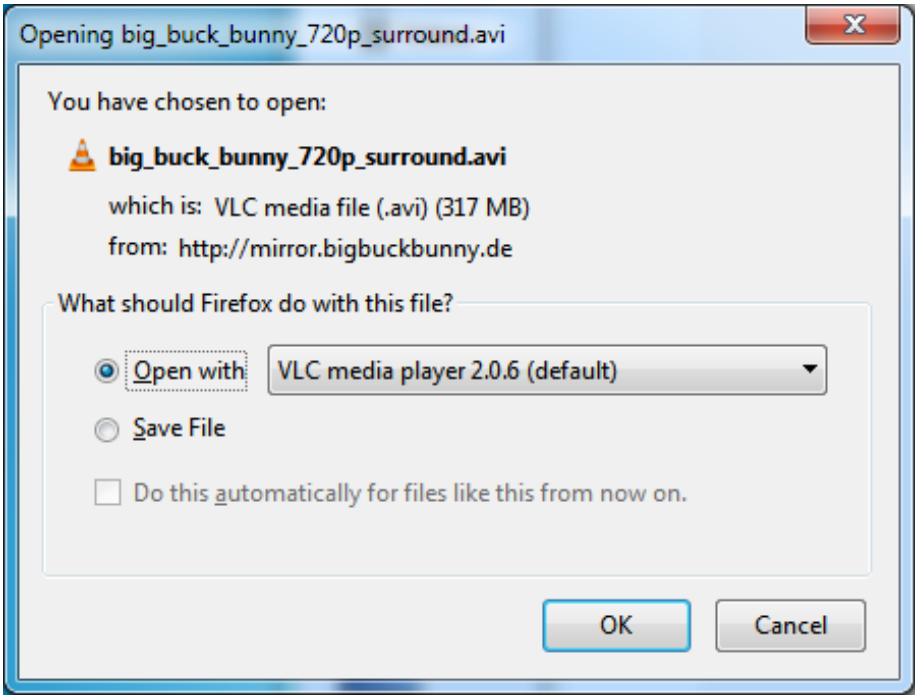
if your video is not sized for 320 wide, you may need to add a -zoom after -framedrop so that it will resize - note that this is quite taxing for the Pi, so it may result in a choppy or mis-synced video!


- Try a different -vo driver (-vo help for a list)
or try -framedrop!
Big Buck BUNNY
A: 28.0 V: 27.1 A-V: 0.875 ct: 2.681 0/ 0 64%
adafruit
Connector on reverse!

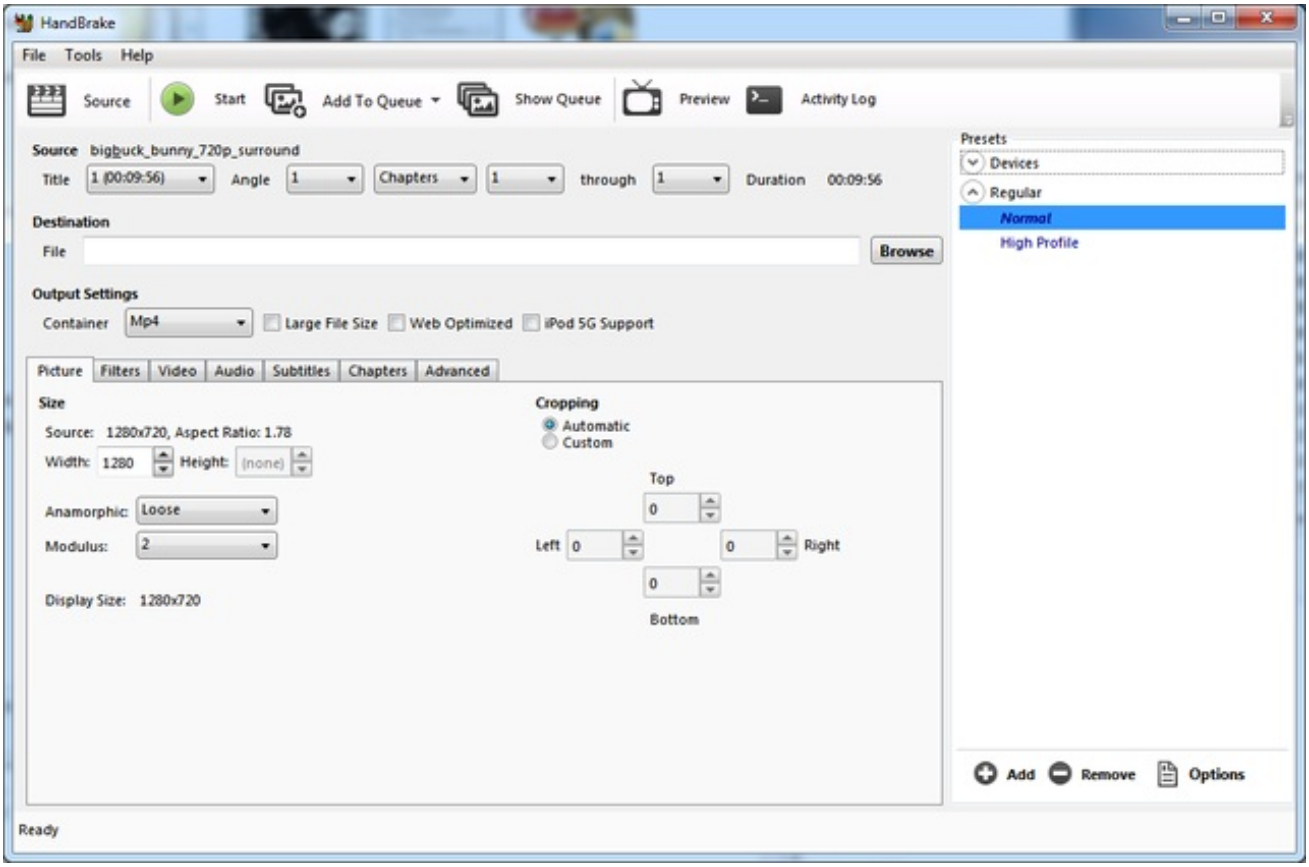
Converting/Resizing Videos

It's possible to play full length videos on the TFT plate, but since the screen is small and the Pi cant use hardware accelleration to play the videos its best to scale them down to 320x240 pixels. This will be easier for the Pi to play and also save you tons of storage space. For this demo, we'll be using the famous Big Buck Bunny (http://adafru.it/cXS) video, which is creative commons and also very funny!

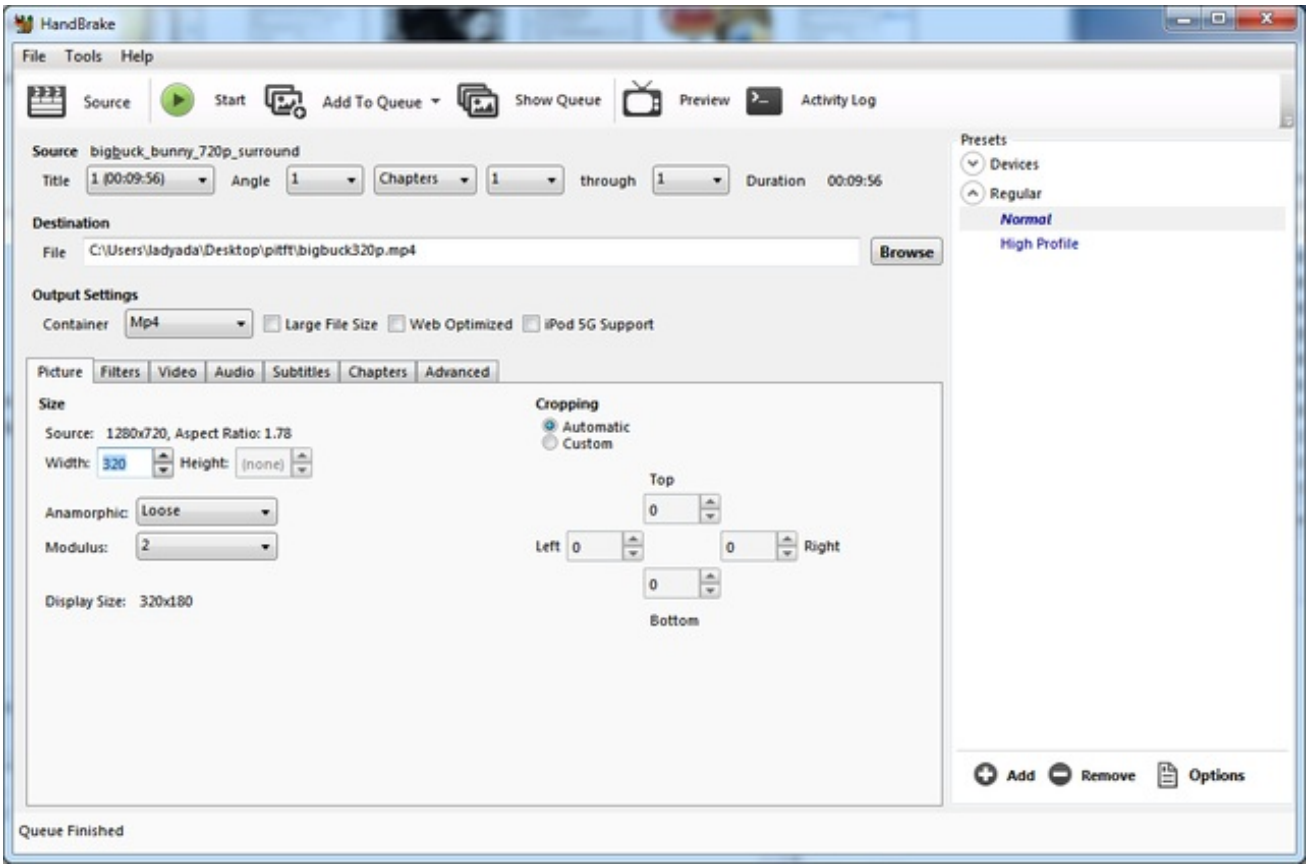
You can download it from the link above, we'll be using the 720p AVI version.



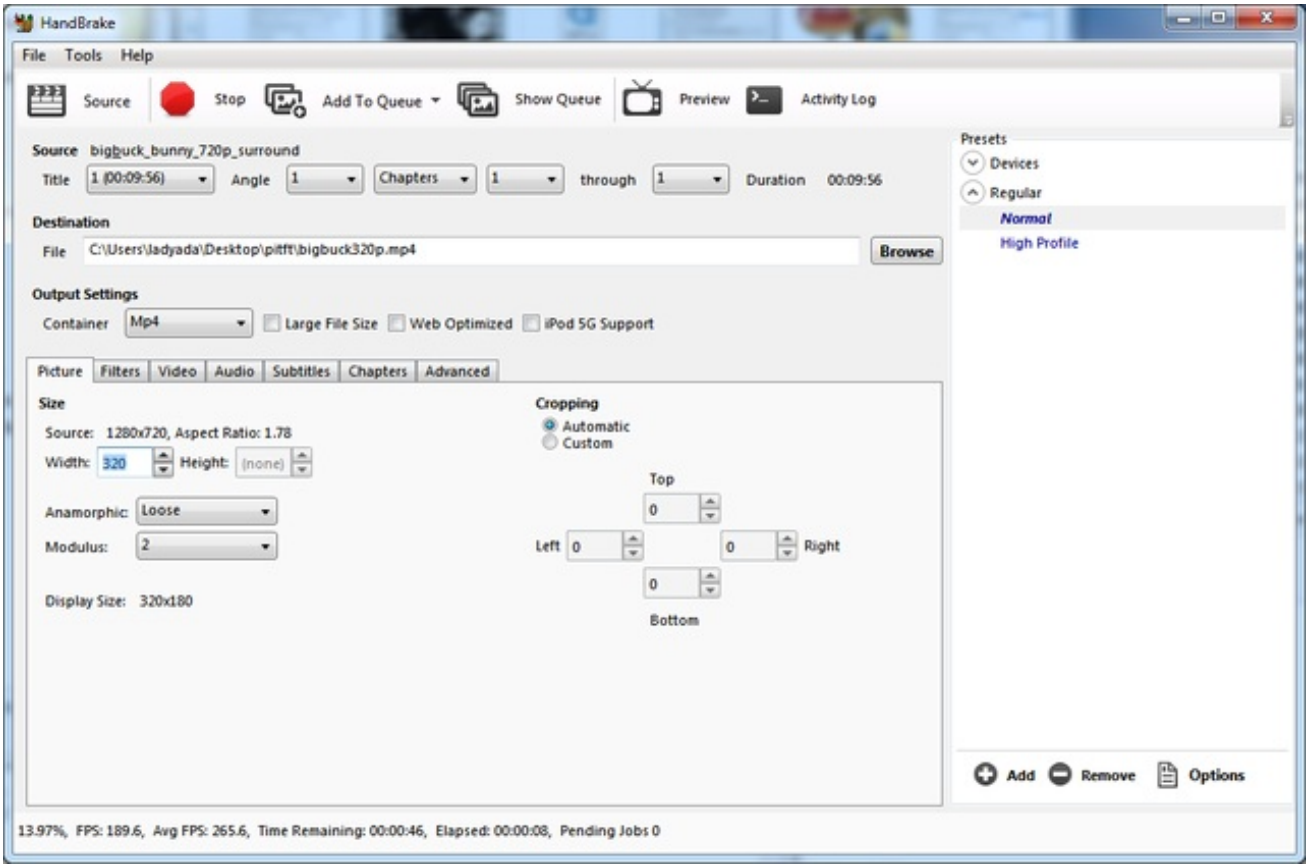
To do the conversion itself, we suggest HandBrake (http://adafru.it/cXT) which works great and is open source so it runs on all operating systems! Download and install from the link. Then run the installed application and open up the AVI file from before. The app will pre-fill a bunch of information about it.



Under **Destination** click **Browse...** to select a new MP4 file to save. Then under **Picture** change the **Width** to 320 (the height will be auto-calculated)



Click **START** to begin the conversion, it will take a minute or two.

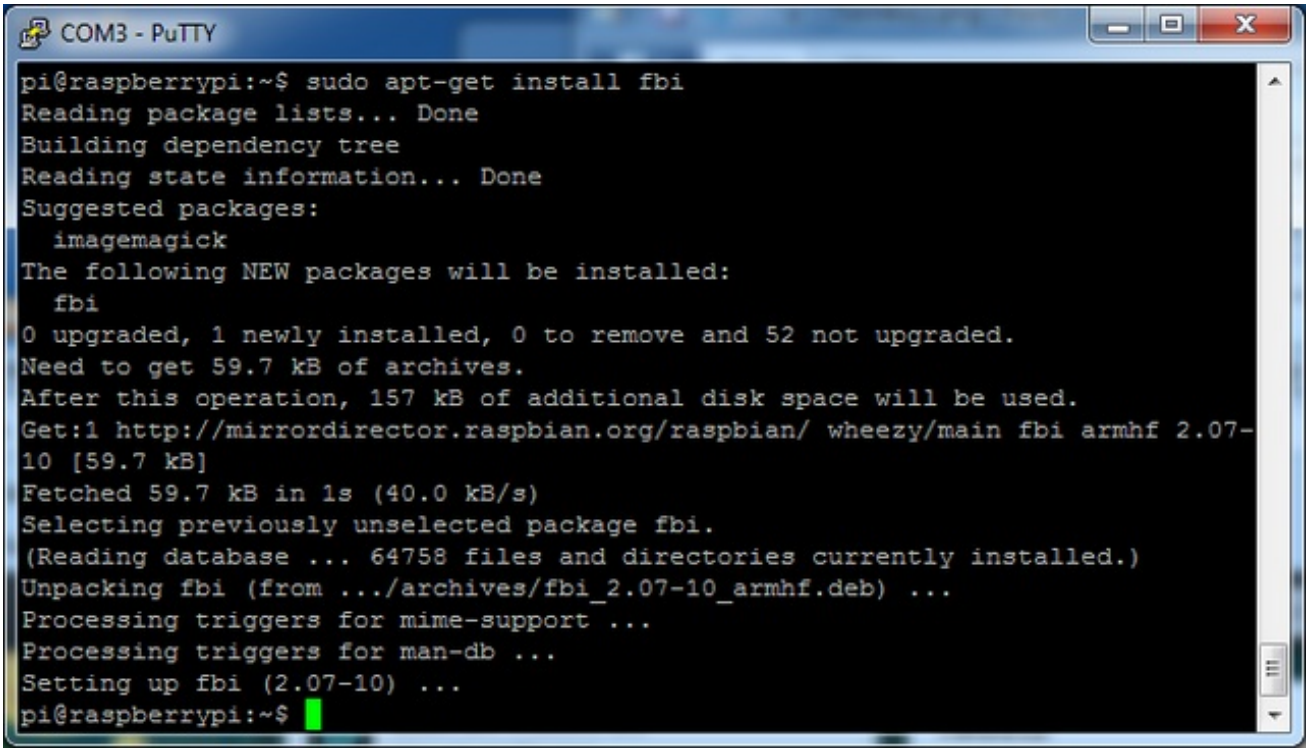


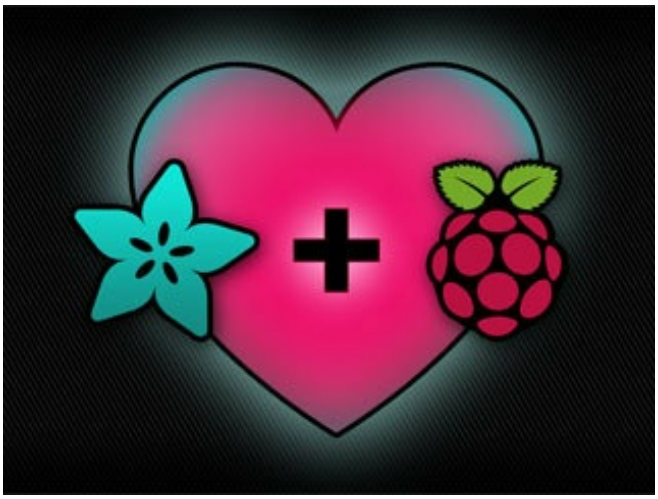
That's it! You now have a smaller file. Don't forget to play it on your computer to make sure it plays right before copying it to your Pi

Displaying Images

You can display every day images such as GIFs, JPGs, BMPs, etc on the screen. To do this we'll install **fbi** which is the **frame buffer image** viewer (not to be confused with the FBI agency!)

sudo apt-get install fbi will install it


COM3 - PuTTY
pi@raspberrypi:~$ sudo apt-get install fbi
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
imagemagick
The following NEW packages will be installed:
fbi
0 upgraded, 1 newly installed, 0 to remove and 52 not upgraded.
Need to get 59.7 kB of archives.
After this operation, 157 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main fbi armhf 2.07-10 [59.7 kB]
Fetched 59.7 kB in 1s (40.0 kB/s)
Selecting previously unselected package fbi.
(Reading database ... 64758 files and directories currently installed.)
Unpacking fbi (from .../archives/fbi_2.07-10_armhf.deb) ...
Processing triggers for mime-support ...
Processing triggers for man-db ...
Setting up fbi (2.07-10) ...
pi@raspberrypi:~$



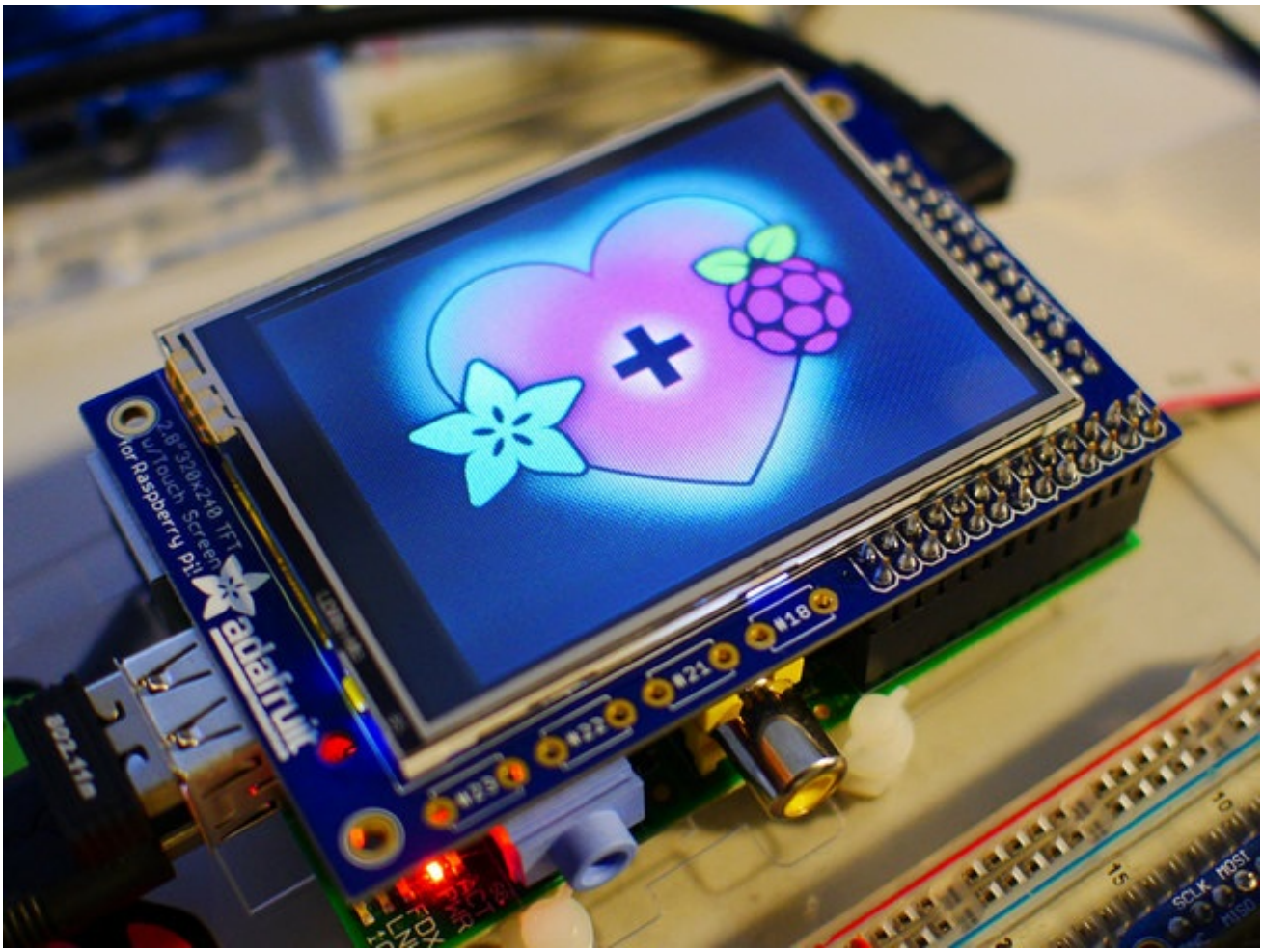
Grab our lovely wallpaper with

> **wget http://adafruit-download.s3.amazonaws.com/adapiluv320x240.jpg** (http://adafru.it/cXU)

and view it with

```
sudo fbi -T 2 -d /dev/fb1 -noverbose -a adapiluv320x240.jpg
```

That's it!



Backlight Control

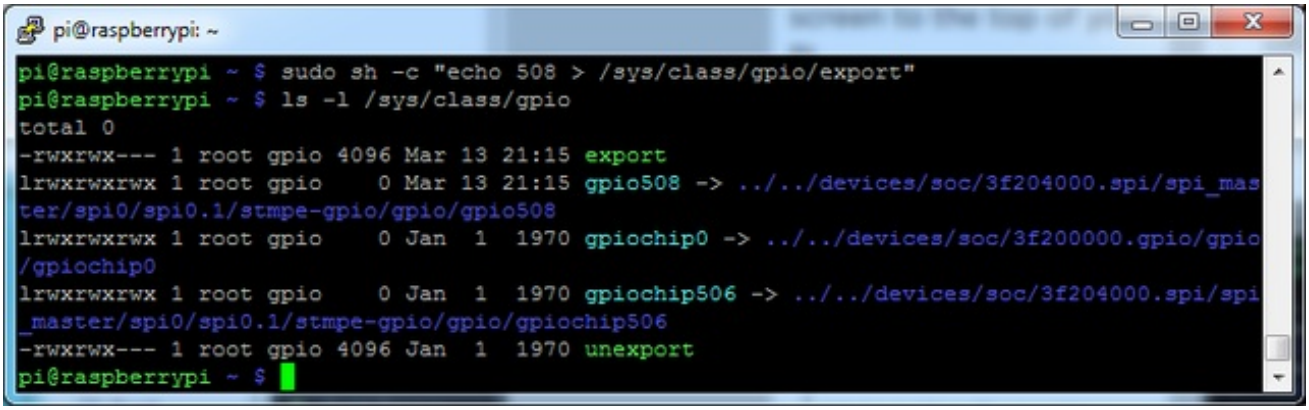
Controlling the backlight

There's a 4-LED backlight on the TFT and it draws ~75mA at all times. There might be times you'd like to save some power and turn off the backlight. The screen and touchplate will still work, you just can't see anything. We designed the board with the STMPE610 touchscreen controller which has 2 extra GPIO and tied one of them to the transistor that controls the backlight. You can use the command line to control the backlight.

Start by getting access to the GPIO by making a device link

```
sudo sh -c "echo 508 > /sys/class/gpio/export"
ls -l /sys/class/gpio
```

Please note, previous versions of the PiTFT kernel had the STMPE GPIO pin '252' not '508' so just substitue that number


pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo sh -c "echo 508 > /sys/class/gpio/export"
pi@raspberrypi ~ $ ls -l /sys/class/gpio
total 0
-rwxrwx--- 1 root gpio 4096 Mar 13 21:15 export
lrwxrwxrwx 1 root gpio 0 Mar 13 21:15 gpio508 -> ../../devices/soc/3f204000.spi/spi_master/spi0/spi0.1/stmpe-gpio/gpio/gpio508
lrwxrwxrwx 1 root gpio 0 Jan 1 1970 gpiochip0 -> ../../devices/soc/3f200000.gpio/gpio/gpiochip0
lrwxrwxrwx 1 root gpio 0 Jan 1 1970 gpiochip506 -> ../../devices/soc/3f204000.spi/spi_master/spi0/spi0.1/stmpe-gpio/gpio/gpiochip506
-rwxrwx--- 1 root gpio 4096 Jan 1 1970 unexport
pi@raspberrypi ~ $

Once you verify that you see GPIO #508, then you can set it to an output, this will turn off the display since it will output 0 by default

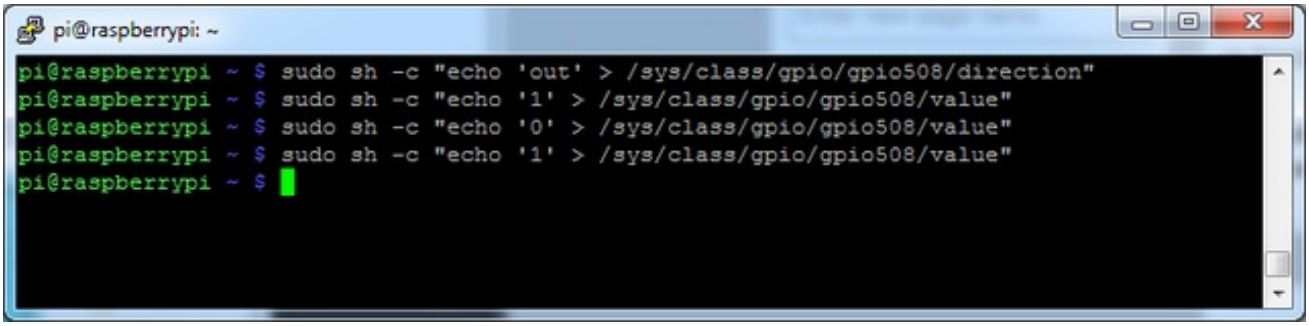
```
sudo sh -c "echo 'out' > /sys/class/gpio/gpio508/direction"
```

Then turn the display back on with

```
sudo sh -c "echo '1' > /sys/class/gpio/gpio508/value"
```

or back off

```
sudo sh -c "echo '0' > /sys/class/gpio/gpio508/value"
```


pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo sh -c "echo 'out' > /sys/class/gpio/gpio508/direction"
pi@raspberrypi ~ $ sudo sh -c "echo '1' > /sys/class/gpio/gpio508/value"
pi@raspberrypi ~ $ sudo sh -c "echo '0' > /sys/class/gpio/gpio508/value"
pi@raspberrypi ~ $ sudo sh -c "echo '1' > /sys/class/gpio/gpio508/value"
pi@raspberrypi ~ $

Extras!

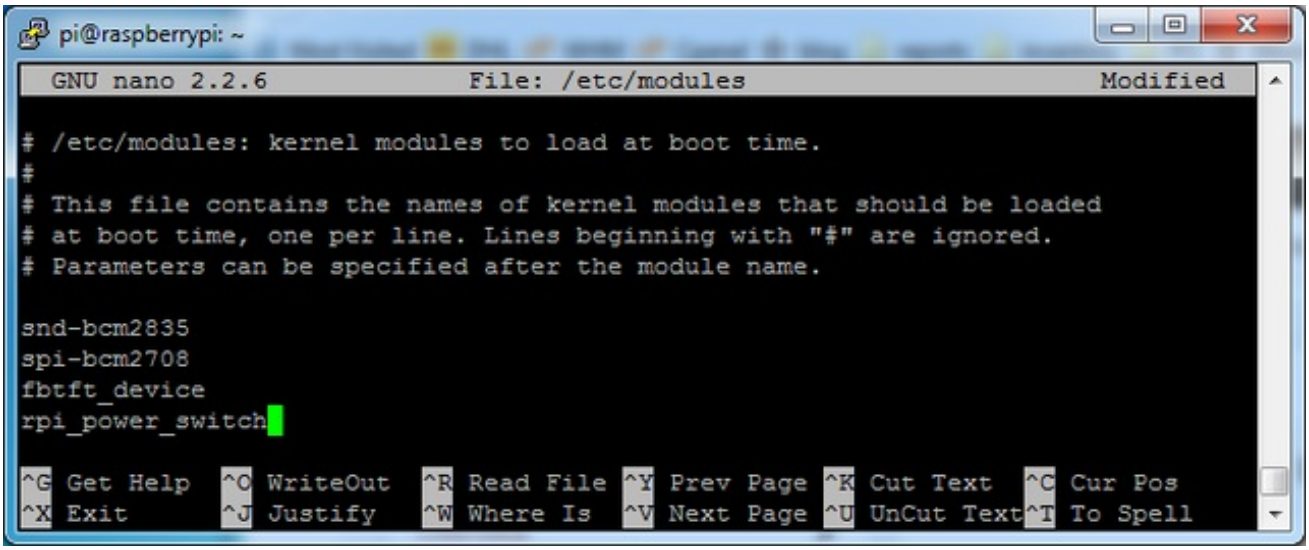
Tactile switch as power button

Its a good idea to safely turn off your Pi with a good **sudo shutdown -h now** but that often means pulling out a keyboard or connecting to the console. With our kernel we added a cool module that will let you turn any GPIO into a power button. Since there's a couple of tactile switches right there on the front, lets turn one into a power button. Press once to properly turn off the pi, press again to start it up. Isn't that nice?

We'll be using GPIO #23, the left-most button on a PiTFT 2.8", on the 2.4" HAT, #16 is a good choice since its on a tactile button. But, you can use any GPIO you want, really!

You will have to grab a pack of slim tactile switches (http://adafru.it/1489) or otherwise solder in a button

Add **rpi_power_switch** to /**etc/modules** and save


pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/modules Modified
/etc/modules: kernel modules to load at boot time.
#
This file contains the names of kernel modules that should be loaded
at boot time, one per line. Lines beginning with "#" are ignored.
Parameters can be specified after the module name.
snd-bcm2835
spi-bcm2708
fbtft_device
rpi_power_switch
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

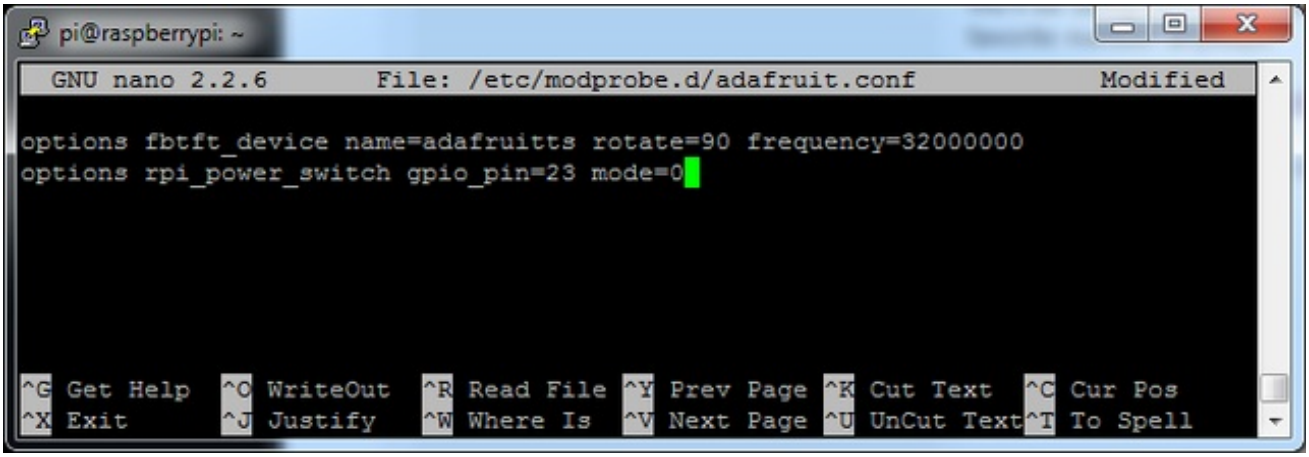
Now create a new conf file or edit our existing one with

> **sudo nano /etc/modprobe.d/adafruit.conf**

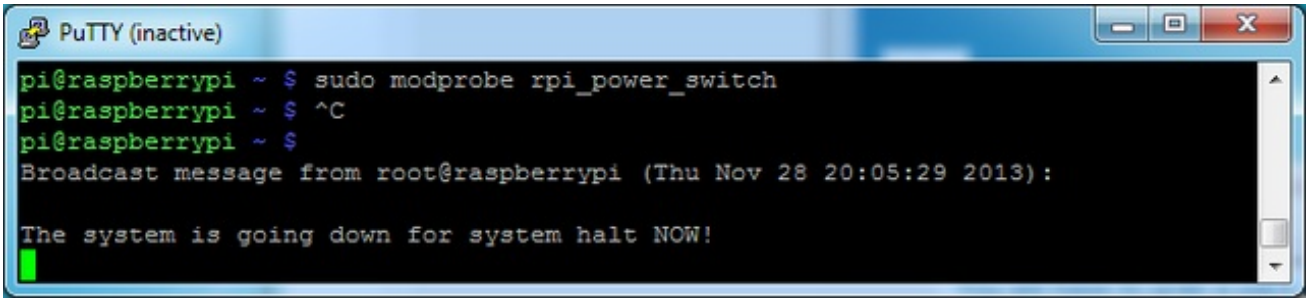
and enter in the line

> **options rpi_power_switch gpio_pin=23 mode=0**

Of course, change the **gpio_pin** setting to some other # if you wish. **mode=0** means its a pushbutton *not* a switch. If you happen to install an on/off switch, use **mode=1**


pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/modprobe.d/adafruit.conf Modified
options fbtft_device name=adafruitts rotate=90 frequency=32000000
options rpi_power_switch gpio_pin=23 mode=0
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

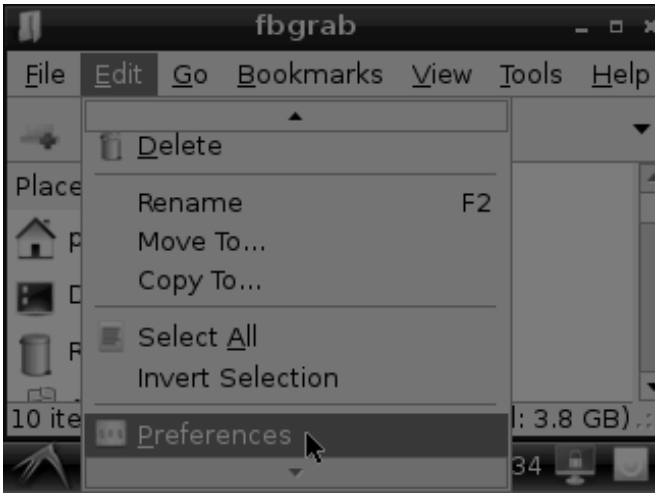
To make it active immediately run **sudo modprobe rpi_power_switch**


PuTTY (inactive)
pi@raspberrypi ~ $ sudo modprobe rpi_power_switch
pi@raspberrypi ~ $ ^C
pi@raspberrypi ~ $
Broadcast message from root@raspberrypi (Thu Nov 28 20:05:29 2013):
The system is going down for system halt NOW!

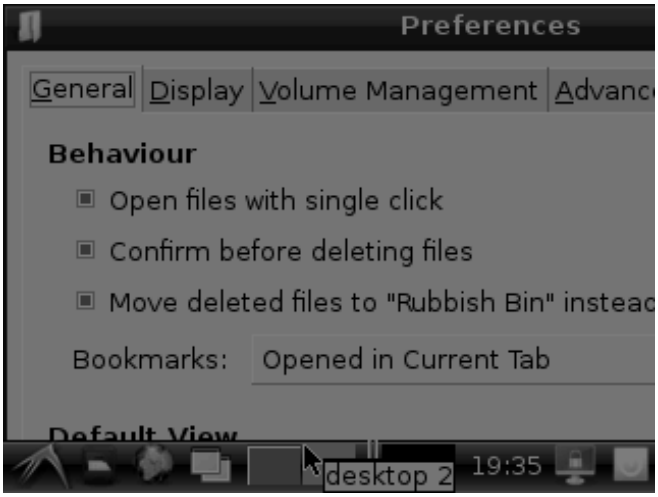
Making it easier to click icons in X

If you want to double-click on icons to launch something in X you may find it annoying to get it to work right. In LXDE you can simply set it up so that you only need to single click instead of double. From LXDE launch the file manager (sorry these pix are grayscale, still figuring out how to screenshot the framebuffer!)

Then under the **Edit** menu, select **Preferences**



Then select **Open files with single click** and close the window (you'll need to drag it over to get to the X button



Boot to X Windows on PiTFT

To enable booting straight to X windows on the PiTFT follow the steps below. First make sure a display configuration which would conflict is **not** present by executing in a terminal on the Pi:

```
sudo mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf ~
```

Don't worry if the command fails with an error that the file doesn't exist. This failure is normal and should happen on a good PiTFT install. You can ignore it and move on.

Next run the command below to open the nano text editor as root and create the file **/usr/share/X11/xorg.conf.d/99-pitft.conf**:

```
sudo nano /usr/share/X11/xorg.conf.d/99-pitft.conf
```

When the editor loads to a blank file, copy in the text below:

```
Section "Device"
  Identifier "Adafruit PiTFT"
  Driver "fbdev"
  Option "fbdev" "/dev/fb1"
EndSection
```

Then save the file by pressing **Ctrl-O** and then **enter**, and finally exit by pressing **Ctrl-X**.

The step above will create a configuration file which tells X windows that it should use the PiTFT framebuffer (located at /dev/fb1) by default when it runs.

At this point you can use the raspi-config tool to enable booting to desktop just like normal on the Pi. Run the following command:

```
sudo raspi-config
```

Then pick the **Enable Boot to Desktop/Scratch** option and choose if you want to boot to the console, desktop, or scratch environment. After exiting the tool and rebooting you should see the Pi load X windows on the PiTFT after (be patient it can take around 30 seconds to load).

If you want to disable booting to X, just use the raspi-config command again to choose the console boot option.

Gesture Input

With the PiTFT touchscreen and xstroke (http://adafru.it/dD0) you can enter text in applications by drawing simple character gestures on the screen! Check out the video below for a short demonstration and overview of gesture input with xstroke:

Installation

Unfortunately xstroke hasn't been actively maintained for a few years so there isn't a binary package you can directly install. However compiling the tool is straightforward and easy with the steps below. Credit for these installation steps goes to mwilliams03 at ozzmaker.com (http://adafru.it/dD1).

First install a few dependencies by opening a command window on the Pi and executing:

```
sudo apt-get -y install build-essential libxft-dev libxpm-dev libxtst-dev
```

Now download, compile, and install xstroke by executing:

```
cd ~
wget http://mirror.egtvedt.no/avr32linux.org/twiki/pub/Main/XStroke/xstroke-0.6.tar.gz
tar xfv xstroke-0.6.tar.gz
cd xstroke-0.6
./configure
sed -i '/^X_LIBS = / s/$/ -lXrender -lX11 -lXext -ldl/' Makefile
make
sudo make install
```

If the commands above execute successfully xstroke should be installed. If you see an error message, carefully check the dependencies above were installed and try again.

Once xstroke is installed you will want to add a couple menu shortcuts to start and stop xstroke. Execute the following commands to install these shortcuts:

```
wget https://github.com/adafruit/PiTFT_Extras/raw/master/xstroke.desktop
wget https://github.com/adafruit/PiTFT_Extras/raw/master/xstrokekill.desktop
sudo cp xstroke*.desktop /usr/share/applications/
```

Usage

To use xstroke I highly recommend using a plastic stylus instead of your finger. Also calibrate the touchscreen for X-Windows (http://adafru.it/dD2) so you have the best control over the cursor

possible.

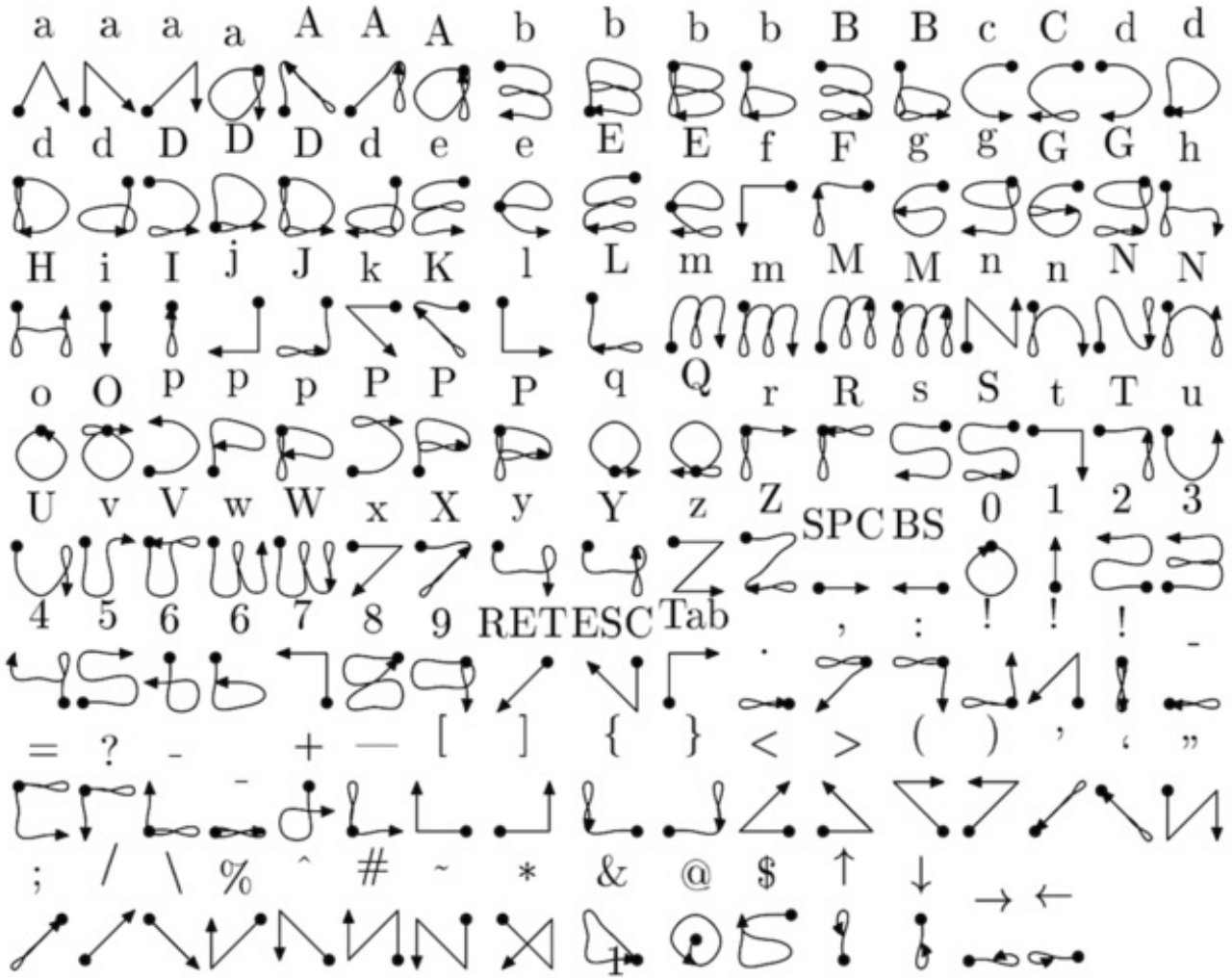
Don't use a ballpoint pen or sharp metal stylus as it could scratch or damage the touchscreen!

Start X-Windows on the PiTFT and open the LXDE menu by clicking the icon in the lower left corner. Scroll up to the **Accessories** menu at the top and notice the new **XStroke** and **XStroke Kill** commands.

Click the **XStroke** menu option to start xstroke. You should see a small pencil icon appear on the bottom right side of the screen. The pencil icon means xstroke is running, however by default it's not yet looking for gesture input.

Open an application that takes text input, such as LXTerminal. To enable gesture input click the xstroke pencil icon. You should see the pencil turn green and the text 'abc' written over top of the icon. You might need to click the icon a few times to get the click to register in the right spot.

When xstroke is looking for gesture input you can drag the mouse cursor in a gesture anywhere on the screen to send specific key strokes. Here's a picture of the possible gestures you can send:


a a a a A A A b b b b B B c C d d
d d D D D d e e E E f F g g G G h
H i I j J k K l L m m M M n n N N
o O p p p P P P q Q r R s S t T u
U v V w W x X y Y z Z SPC BS 0 1 2 3
4 5 6 6 7 8 9 RET ESC Tab , : ! ! !
= ? - + — [ ] { } < > ( ) ’ ‘ ”
; / \ % ^ # ~ * & @ $ ↑ ↓ → ←

(credit to Carl Worth for the image above)

To draw a gesture from the above image, press anywhere on the screen, start from the circle in the gesture, and follow the gesture pattern towards the arrow. As you draw a gesture you should see a blue line displayed that shows what you've drawn. Lift up the stylus when you get to the end of the gesture at the arrow. If xstroke recognizes the gesture it will send the appropriate key press to the active window. Try drawing a few characters from the image above to get the hang of writing gestures.

A few very useful gestures are backspace (which deletes a character), return/enter, and space. To draw a backspace gesture just draw a line going from the right side of the screen to the left side. The gesture for return/enter is a diagonal line from the top right to bottom left. Finally a space is a straight line from the left to the right.

Note that when xstroke is looking for gestures you might not be able to click or control the cursor as you normally would expect. To stop xstroke's gesture recognition carefully press the xstroke pencil icon again until the 'abc' text disappears. I've found this process can be a little finicky as the icon is

very small and any movement will be interpreted as a gesture. Use a light touch and try a few times to click the icon.

If you get stuck completely and can't disable xstroke by clicking the icon, connect to the Raspberry Pi in a terminal/SSH connection and run 'killall xstroke' (without quotes) to force xstroke to quit. The normal way to stop xstroke is to navigate to the **Accessories** -> **XStroke Kill** command, but you might not be able to do that if xstroke is listening for gesture input.

Have fun using xstroke to control your Pi by writing gestures on the PiTFT screen!

FAQ

My PiTFT used to work, now it doesn't!

Did you do an `apt-get upgrade` or `rpi-update`? This command will blow away our PiTFT kernel which means that you will no longer have PiTFT support, you will have to redo the **easy-install** steps to reinstall the kernel.

If you had already made a working PiTFT setup, you may be able to reinstall the Adafruit kernel like so:

```
sudo apt-get install raspberrypi-bootloader=1.20150309-1
```

If it tells you that the latest version is already installed, try this instead:

```
sudo apt-get install --reinstall raspberrypi-bootloader=1.20150309-1
```

...you can check here (http://adafru.it/eUK) and substitute the most recent version you see in the `=1.20150309-1` part.

My PiTFT works for a bit and then I get a black screen with a short line of white pixels in one corner

Sounds like you tried to configure your Pi to 'boot straight to X', that is, start up the graphics interface on boot. This doesn't work by default because the Pi operating system is not expecting a PiTFT so it boots to the HDMI output. See below for how to set up your Pi to boot to X on the PiTFT

To 'fix' this, you can either connect an HDMI monitor, then in a terminal window run **sudo raspi-config** and configure the Pi to boot to the command line not X! If you do not have an HDMI monitor, you can also try a console cable

How can I force the Pi to bring up X on the HDMI/TV monitor?

Use the **fb0** framebuffer when you want to display stuff on the HDMI/TV display, for example:
FRAMEBUFFER=/dev/fb0 startx
will use the HDMI/TV framebuffer for X windows instead of the PiTFT

That doesn't work! I can't get X on HDMI!

If both
FRAMEBUFFER=/dev/fb0 startx &
and
FRAMEBUFFER=/dev/fb1 startx &
wind up showing the GUI on your PiTFT, enter the following instruction from the command line:

```
sudo mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf ~
```

I'm tring to run startx and I get FATAL: Module g2d_23 not found.

don't forget you have to remove the turbo file!

sudo mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf ~

Why doesn't the tactile button on GPIO #21 work?

On some older PiTFTs we had one of the buttons labeled #21 - that's the original RasPi name for that pin. If you're using a V2 (chance is, you are!) that is now called #27.

All the PiTFT's we ship now have the button labeled #21 and #27

I want better performance and faster updates!

You can change the SPI frequency (overclock the display) by editing /**boot/config.txt** and changing the **dtoverlay** options line to:

> **dtoverlay=pitft28r,rotate=90,speed=62000000,fps=25**

Or whatever you like for speed, rotation, and frames-per-second. BUT, here's the thing, the Pi only supports a *fixed number* of SPI frequencies. So tweaking the number a little won't do anything. The kernel will round the number to the closest value. You will always get frequencies that are 250MHz divided by an even number. Here's the only SPI frequencies this kernel supports

- 15,625,000 (a.k.a 16000000 = 16 MHz)
- 17,857,142 (a.k.a. 18000000 = 18 MHz)
- 20,833,333 (a.k.a 21000000 = 21 MHz)
- 25,000,000 (= 25 MHz)
- 31,250,000 (a.k.a 32000000 = 32MHz)
- 41,666,666 (a.k.a 42000000 = 42MHz)
- 62,500,000 (a.k.a 62000000 = 62MHz)

So if you put in 48000000 for the speed, you won't actually get 48MHz, you'll actually only get about 42MHz because it gets rounded down. We tested this display nicely with 32MHz and we suggest that. But you can put in 42MHz or even try 62MHz and it will update faster

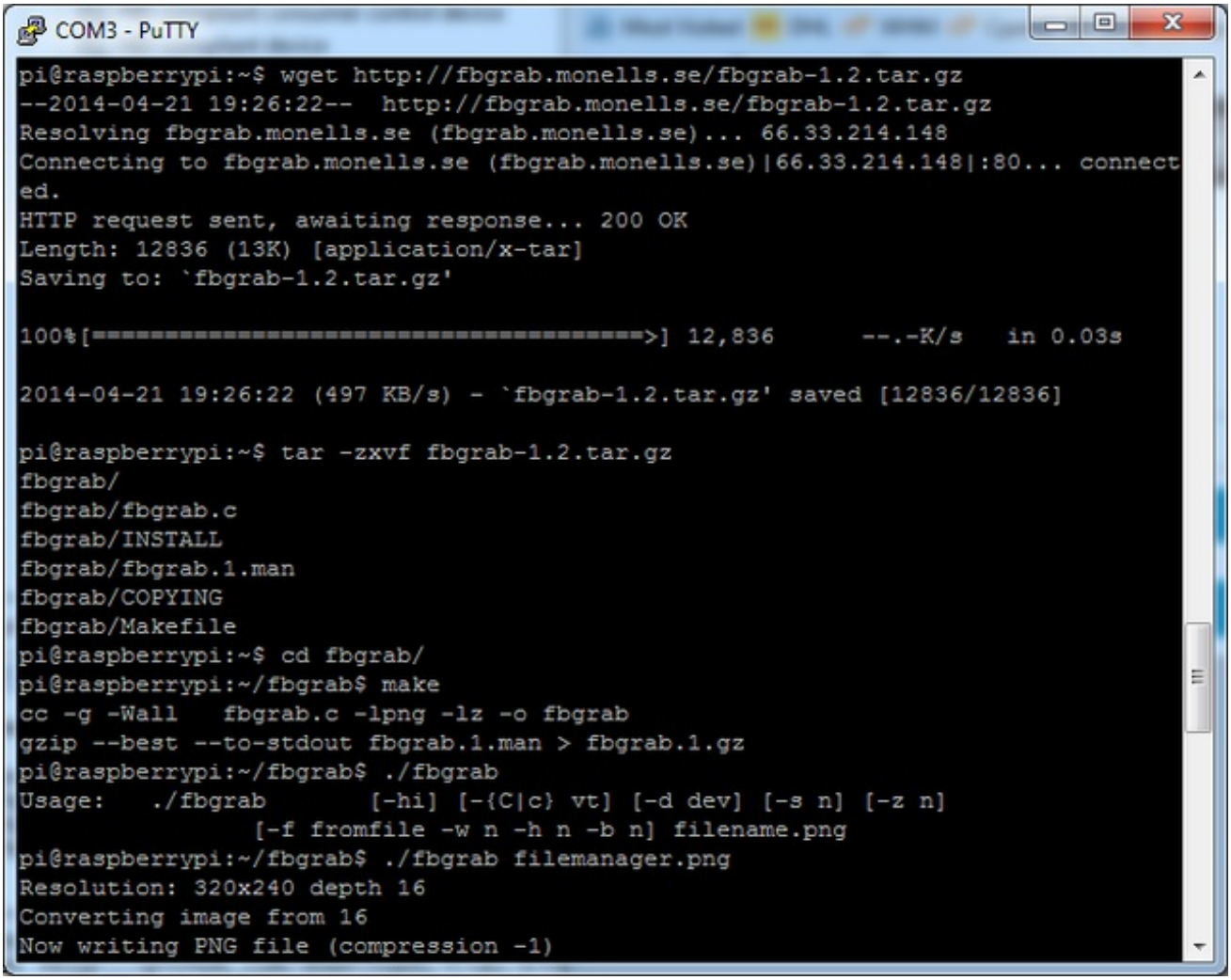
You can tweak fps (frames per second) from 20 to 60 and frequency up to 62MHz for tradeoffs in performance and speed. Reboot after each edit to make sure the settings are loaded properly. There's a trade off that if you ask for higher FPS you're going to load the kernel more because it's trying to keep the display updated.

How can I take screenshots of the little screen?

We took the screenshots for this tutorial with (http://adafru.it/diV)**fbgra (http://adafru.it/diV)b (http://adafru.it/diV)**

> wget http://fbgrab.monells.se/fbgrab-1.2.tar.gz (http://adafru.it/diW)

```
tar -zxvf fbgrab*gz
cd fbgrab/
make
./fbgrab screenshot.png
```

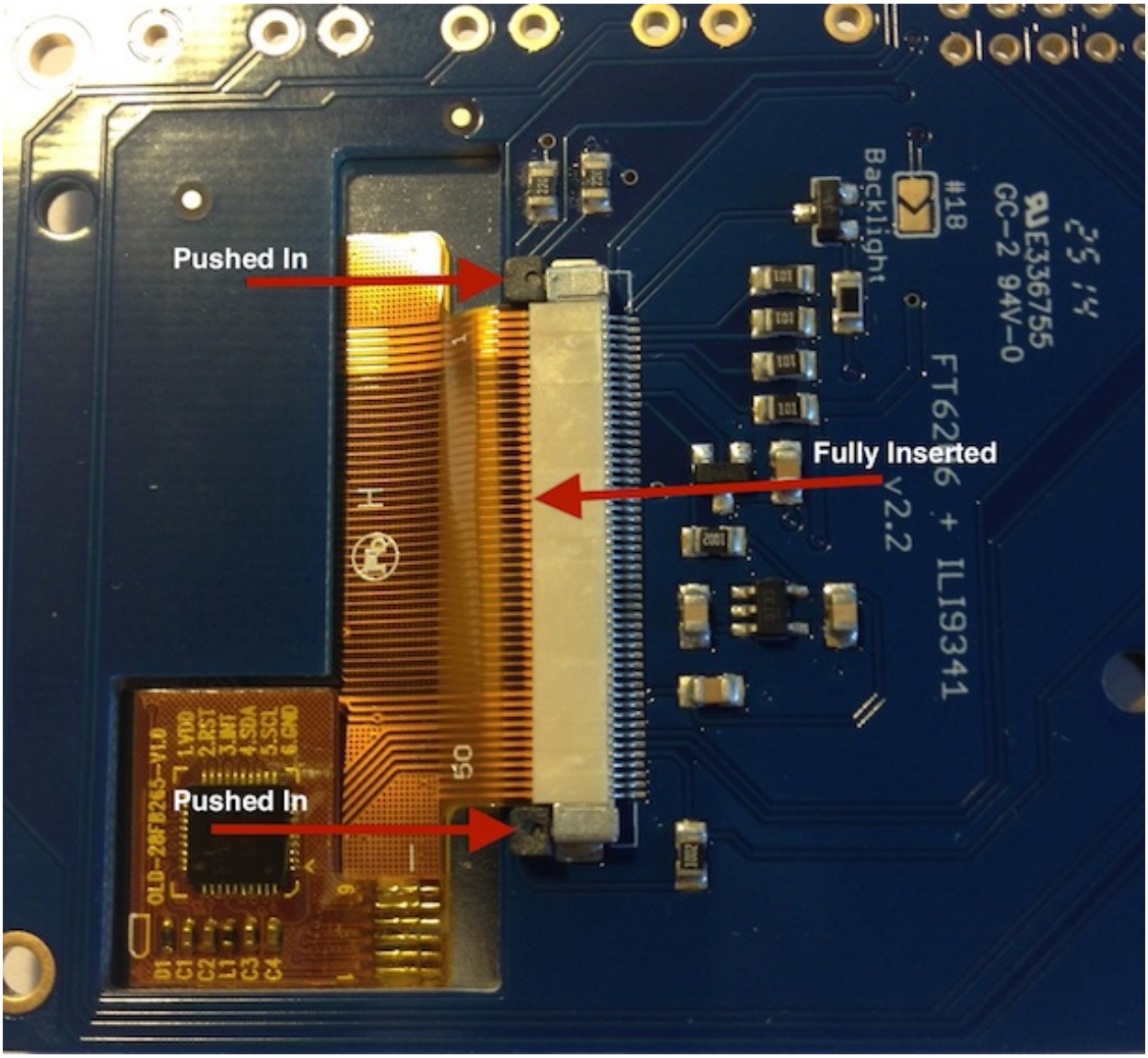


How do I automatically boot to X windows on the PiTFT?

Once you have a PiTFT installation setup you can add a custom X windows configuration to use the PiTFT by default. Then you can use the normal raspi-config boot to console/desktop options to control if the Pi boots to a console or desktop. See the detailed instructions on this page of the guide (http://adafru.it/e1s) for more information.

My screen isn't working/works erratically/looks funny

Check to make syre that the flat flex cable is fully seated in the connetor and the 'ears' are pushed in to secure it. See the picture for what it should look like:



On my first run of startx I get a window saying "GDBus Error.org.Freedesktop Policy Kit1 Error: Failed Cannot determine user of subject"

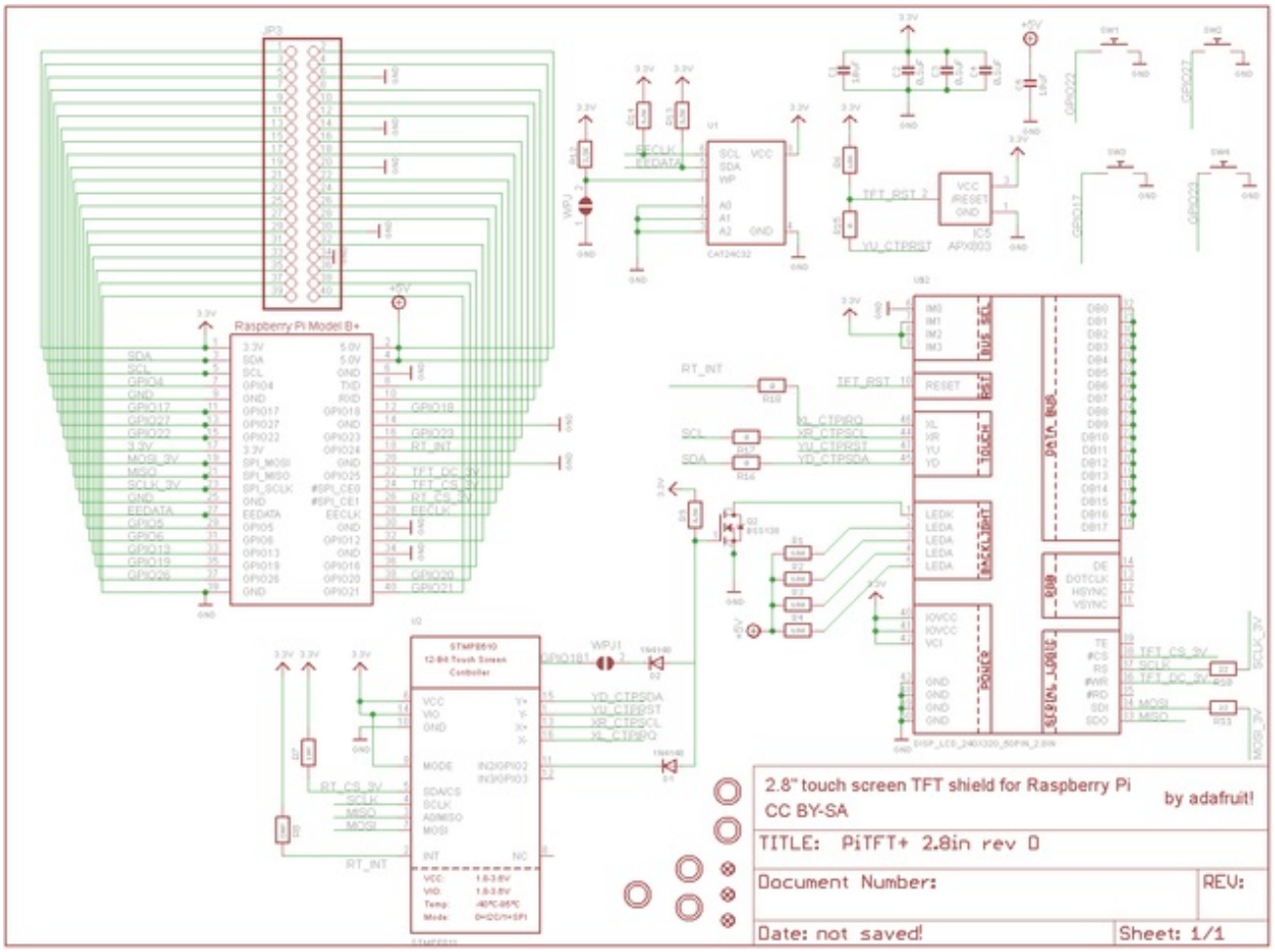
This happens on the Raspberry Pi the first time you run startx, no matter what display. You can just re-start X and it wont appear again.

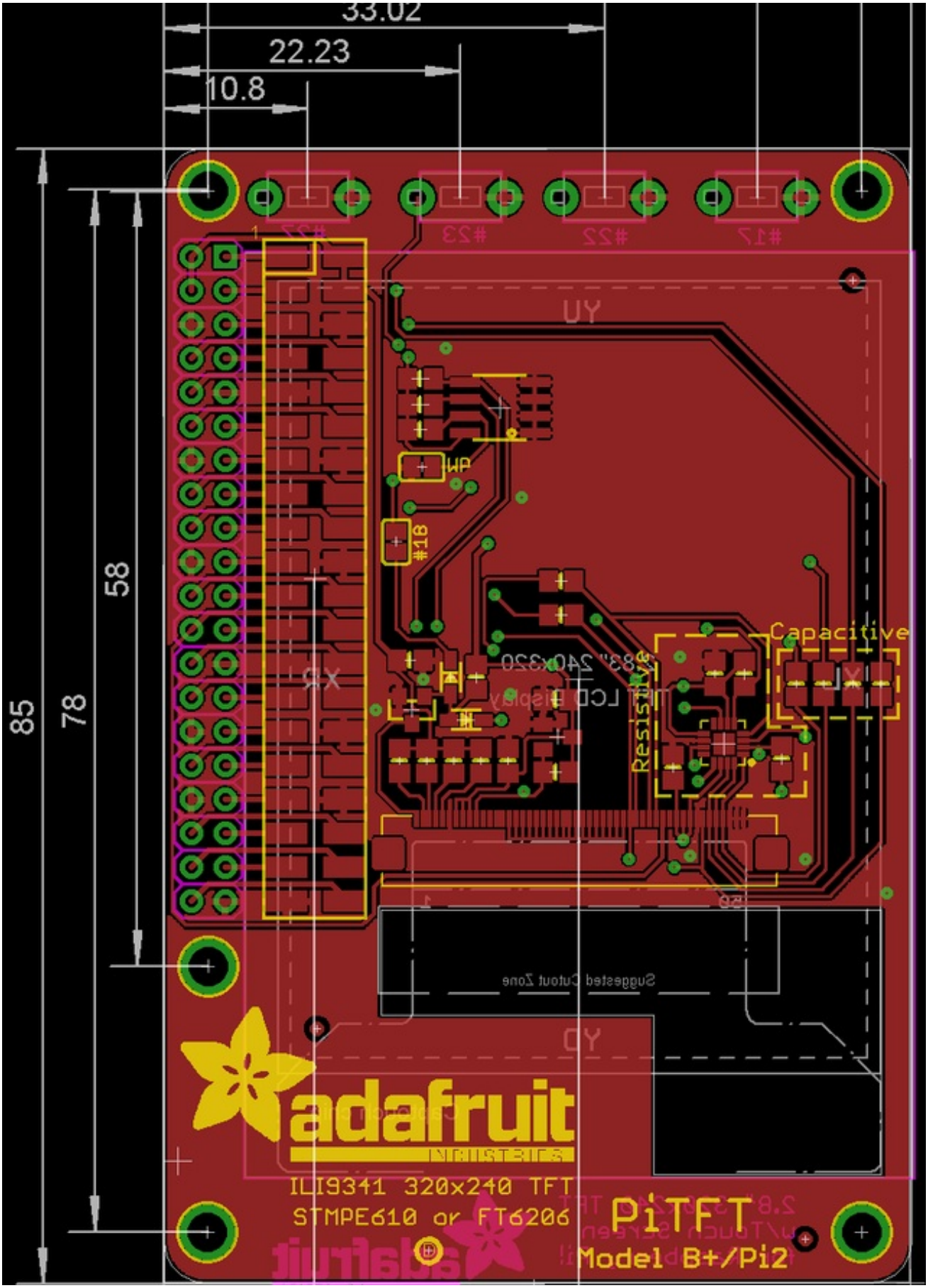
Downloads

- The latest kernel fork that adds all the TFT, touchscreen, and other addons is here on github (http://adafru.it/dcA)
- Datasheet for the 'raw' 2.8" TFT display (http://adafru.it/d4m)

Schematic & Layout

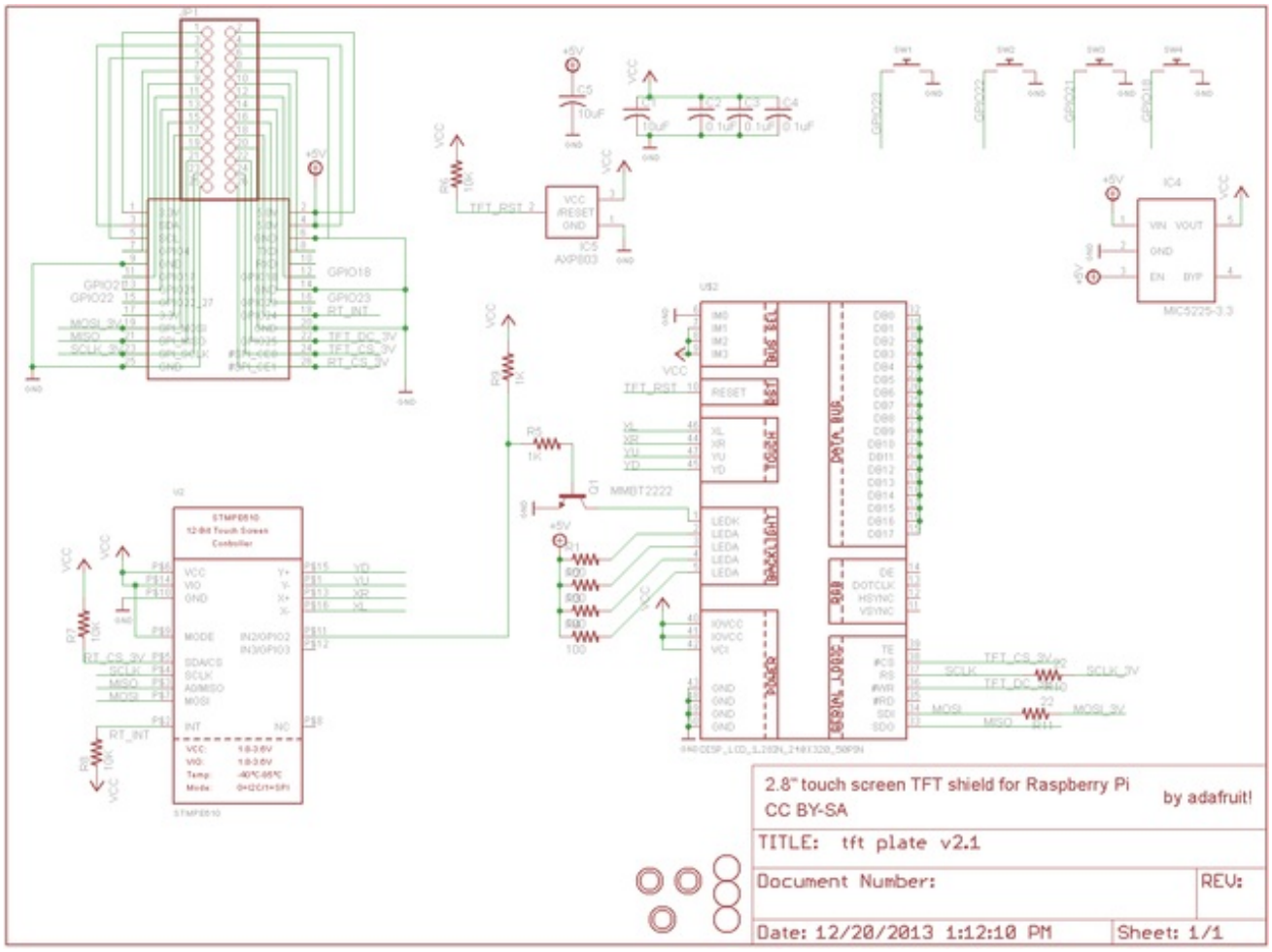
For the Pi B+ & Pi 2 version (2x20 header)

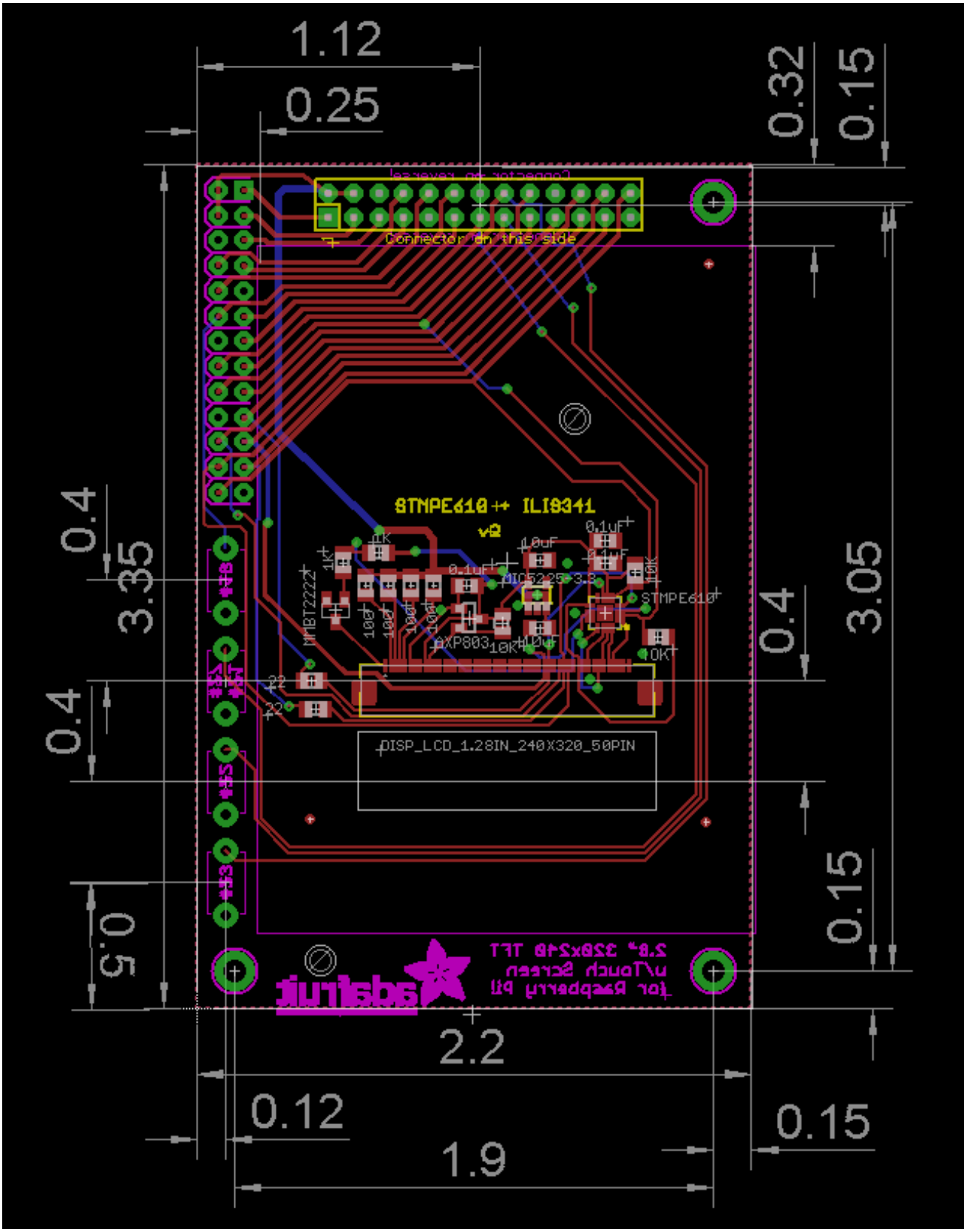



33.02
22.23
10.8
85
78
58
#22
#23
#22
#17
UY
WP
#18
Capacitive
Resistive
X2
Suggested Cutout Zone
YD
adafruit
INDUSTRIES
ILI9341 320x240 TFT
STMPE610 or FT6206
PiTFT
Model B+/Pi2

Schematic & Layout

For the Original Pi 1 version (2x13 header)




1.12
0.25
0.32
0.15
Connector on this side
STMPE610 + ILI9341
v2
0.4
3.35
0.4
3.05
0.4
0.4
0.5
0.15
DISP_LCD_1.28IN_240X320_50PIN
2.2
0.12
1.9
0.15